COLOR IMAGE PROCESSING USING PRINCIPAL COMPONENT ANALYSIS

By

Arash Abadpour

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
SHARIF UNIVERSITY OF TECHNOLOGY
TEHRAN, IRAN
JULY 2005

© Copyright by Arash Abadpour, 2005

SHARIF UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF
MATHEMATICS SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Color Image Processing Using Principal Component Analysis**" by **Arash Abadpour** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: July 2005

Supervisor: _____
Shohreh Kasaei & Amir Daneshgar

Readers: _____

_____

SHARIF UNIVERSITY OF TECHNOLOGY

Date: **July 2005**

Author: **Arash Abadpour**

Title: **Color Image Processing Using Principal Component Analysis**

Department: **Mathematics Science**

Degree: **M.Sc.** Convocation: **June** Year: **2005**

Permission is herewith granted to Sharif University of Technology to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.

THE AUTHOR ATTESTS THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED.

To Azadeh.

Table of Contents

Table of Contents **v**

List of Tables **x**

List of Figures **xv**

Abstract **xxxix**

Acknowledgements **xl**

1 Introduction **1**

2 Literature Review **5**

2.1 Color Spaces 6

2.1.1 Formulations 6

2.1.2 Implementation in **CIP** 37

2.2 Color Image Processing 48

2.2.1 Spotting 48

2.2.2 Distance Measures 49

2.2.3 Fuzzification Function 49

2.2.4 Clustering 50

2.2.5 Tree Decomposition 52

2.2.6 Query Region Extraction 53

2.2.7 Segmentation 54

2.2.8 Principal Colors 55

2.2.9 Alpha Estimation 56

2.2.10 Compression 57

2.2.11 Watermarking 58

2.2.12 Direct Copy Protection 59

2.2.13 Color Transfer 60
2.2.14 Colorizing 61
2.2.15 Recoloring 67
2.2.16 Content–based Pornography Filtering 71
2.2.17 Skin Detection 77
2.3 Color Image Sequence Processing 80
2.3.1 Segmentation 81
2.3.2 Change Detection 83
2.3.3 Recoloring 85
2.4 Registration 86
2.5 Principal Component Analysis (PCA) 88

3 Proposed Methods **94**
3.1 Mathematics 94
3.1.1 Fuzzy Principal Component Analysis (FPCA) 96
3.1.2 Weighted Powered Sum Minimization 100
3.1.3 General Fuzzy Clustering Algorithm 103
3.1.4 Basis Vector Polarization 106
3.1.5 Block–wise Interpolation 111
3.1.6 Butterworth membership function 117
3.1.7 Eigenimages 120
3.1.8 Uniform Quantized Random Variable 123
3.1.9 Quality Measures 124
3.1.10 Finding the Closest Point to a set of Lines. 125
3.1.11 Finding the Closest Point to a set of Fuzzy Lines. 128
3.2 Spotting 129
3.2.1 Color Spaces 129
3.2.2 Parametric Linear Adaptive Color (PLAC) Descriptor 130
3.2.3 PCA–based Parametric Linear Adaptive Color Descriptor . . . 133
3.2.4 Comparison Method 135
3.3 Distance Functions 137
3.4 Query Region Extraction 143
3.4.1 LPRE–Based Approach 145
3.4.2 LLPRE–Based Approach 146
3.5 Segmentation 158
3.5.1 Bi–tree Decomposition 158
3.5.2 FPCA–based Clustering (FPCAC) 163
3.5.3 Agent–based FPCAC 173

3.5.4 Principal Colors . . . 185
3.6 Compression . . . 190
3.6.1 Up–Sampling . . . 190
3.6.2 Block–Wise Interpolation . . . 192
3.7 Watermarking . . . 199
3.7.1 Block–Wise Interpolation . . . 202
3.7.2 Clustering–Based Approach . . . 203
3.8 Direct Copy Protection . . . 205
3.8.1 Clustering–Based Approach . . . 206
3.8.2 Block–Based Approach . . . 213
3.9 Color Transfer . . . 218
3.9.1 Colorizing . . . 218
3.9.2 Region–Based Recoloring . . . 232
3.9.3 Clustering–based Recoloring . . . 240
3.10 Image Registration . . . 244
3.10.1 Direct Linear Transformation and Affine Transform . . . 245
3.10.2 Geometrical Estimation . . . 257
3.10.3 Numerical Enhancement . . . 267
3.11 Content–based Pornography Filtering . . . 273
3.11.1 Look–up Table–Based Approach . . . 273
3.11.2 Histogram–based Approach . . . 278
3.11.3 PCA–based Approach . . . 280
3.12 Video Segmentation and Change Detection . . . 286
3.13 Image Sequence Recoloring . . . 293
3.13.1 Proposed Method . . . 293
3.13.2 Computational Cost of the Available Methods for Still Image Recoloring . . . 299

4 Experimental Results and Discussions 308
4.1 Dimension Reduction in Color Fields . . . 309
4.2 Spotting . . . 311
4.3 Distance Measures . . . 317
4.4 Query Region Extraction . . . 321
4.4.1 LPRE–Based Approach . . . 321
4.4.2 LLPRE–based Approach . . . 336
4.5 Alpha Estimation . . . 337
4.6 Segmentation . . . 339
4.6.1 FPCA–based Clustering (FPCAC) . . . 339

4.6.2 Agent–based FPCAC . . . 359
4.6.3 Tree Decomposition . . . 364
4.6.4 Principal Colors . . . 372
4.7 Compression . . . 376
4.7.1 Up–sampling . . . 376
4.7.2 Block–Wise Interpolation . . . 380
4.8 Watermarking . . . 391
4.8.1 Block–Wise Interpolation . . . 391
4.8.2 Clustering–Based Approach . . . 402
4.9 Direct Copy Protection . . . 417
4.9.1 Clustering–Based Approach . . . 417
4.9.2 Block–Based Approach . . . 428
4.10 Color Transfer . . . 436
4.10.1 Colorizing . . . 436
4.10.2 Query Region–Based Recoloring . . . 448
4.10.3 Clustering based Recoloring . . . 454
4.11 Image Registration . . . 458
4.11.1 Geometrical Estimation . . . 460
4.11.2 Numerical Enhancement . . . 465
4.11.3 Control Point Distribution and Distortions . . . 467
4.12 Content–based Pornography Filtering . . . 474
4.12.1 Datasets . . . 474
4.12.2 Look–up Table–Based Approach . . . 479
4.12.3 Histogram–based Approach . . . 492
4.12.4 PCA–based Approach . . . 499
4.13 Video Segmentation and Change Detection . . . 504
4.14 Change Detection in Remote Sensing Images . . . 515
4.15 Image Sequence Recoloring . . . 521

Conclusions **536**

A Color Image Processing (CIP) Toolbox **537**
A.1 Utility functions . . . 538
A.2 Visualization functions . . . 553
A.3 Data files . . . 564
A.4 Methods Not Described in Chapter 3 . . . 565

B Sample Images **573**
B.1 Color Images . . . 574
B.2 Grayscale Images . . . 592

C Computation Cost **599**
C.1 Basic Operations . . . 600
C.1.1 Vector Addition . . . 600
C.1.2 Vector Multiplication . . . 601
C.1.3 Vector Inner Product . . . 601
C.1.4 Matrix Multiplication . . . 602
C.1.5 Convolution . . . 603
C.1.6 Butterworth Fuzzy Membership Function . . . 603
C.2 Statistics . . . 604
C.2.1 Expectation Vector . . . 604
C.2.2 Standard Deviation . . . 605
C.2.3 Maximum and Minimum . . . 606
C.2.4 Histogram (PDF) Estimation . . . 606
C.2.5 CDF Estimation . . . 607
C.2.6 Covariance Matrix . . . 608
C.2.7 SVD (PCA) . . . 609
C.3 PCA–Based Color Image Processing . . . 610
C.3.1 LPRE . . . 610
C.3.2 Normalized LPRE and LPRE–based Homogeneity Criterion . . . 611
C.3.3 Single–Swatch Recoloring . . . 612
C.3.4 Color Space Conversion . . . 613

D Multi-Layer Representation of Grayscale Images **617**
D.1 Introduction . . . 617
D.2 Proposed Method . . . 618
D.3 Experimental Results . . . 626
D.4 Generalization . . . 632

Bibliography **637**

List of Tables

2.1 Normalization factors for the color spaces under investigation. 38
2.2 Naming convention for the color spaces and their 2–D representations. 39
2.3 Naming convention for the color spaces and their 1–D representations. 40
2.4 Color space conversion functions present in **CIP**. 46
2.5 Summary of the available pornography detection works. 77
2.6 Diversity of color spaces utilized for skin detection. 79

3.1 Papers published from this thesis and the corresponding sections. . . 95
3.2 Impairment scales [189]. 231
3.3 Goodness scales [189]. 231
3.4 Some common transforms in pair notation. 256
3.5 Approximate cost of processing one pixel in different recoloring methods. 307

4.1 Numerical results of the proposed dimension reduction method. . . . 310
4.2 Spotting results in PLAC. 312
4.3 Spotting results in PCA–PLAC. 313
4.4 Comparison of results in color spaces, PLAC and PCA–PLAC. (Part I) 314
4.5 Comparison of results in color spaces, PLAC and PCA–PLAC. (Part II) 315
4.6 Query region extraction results using the Euclidean distance in twelve color spaces, Mahalanobis, and LPRE. Part I [μ_r: recal, μ_p: precision] 326
4.7 Query region extraction results using the Euclidean distance in twelve color spaces, Mahalanobis, and LPRE. Part II [μ_r: recal, μ_p: precision] 326

4.8 Segmentation parameter and the computational cost of the proposed (t_1) and the FCM method [60] (t_2). [m: Fuzzyness, t_1: proposed method, t_2: the FCM method [60]] 353
4.9 Results of three different races on the image shown in Figure 4.37–d. [t: Elapsed Time, J: Objective Function Value. i: Integrity.] 362
4.10 Results of the proposed clustering method performed on the image show in Figure 4.37–e with different values of c. [t: Elapsed Time, J: Objective Function Value. i: Integrity. c: Cluster Count.] 362
4.11 Results of the proposed clustering method performed on the images shown in Figure 4.37. [m: Fuzziness. t: Elapsed Time, J: Objective Function Value. i: Integrity.] 366
4.12 Comparison of the proposed method with the FPCAC described in Section 3.5.2. [t: Elapsed time. i: Number of iterations before convergence. 376
4.13 Numerical results and parameters used for the results shown in Figure 4.68. [λ: Compression Ratio, bpp: bit per pixel, n: block count]. . 393
4.14 Some of the attacks performed on the watermarked images to analyze the robustness of the proposed watermarking method. 398
4.15 Comparison of different watermarking methods with the proposed method used in a 512×512 image. –: Not Resistant. $\sim$: Partially Resistant. $\surd$: Completely Resistant. [Abbreviations: Cap: Capacity, G: Grayscale, SCC: Single Color Component, CV: Color Vector, LG: Linear Geometrical Transformation, NLG: Nonlinear Geometrical Transformation, LPO: Linear Point Operations, NLPO:Nonlinear Point Operations, SO: Spatial Domain Operations, EO: Editing Operations, CMP: JPEG Compression]. 416
4.16 Numerical results of proposed algorithm performed on images shown in Figure 4.88. [m: Fuzziness. t_1 Encryption time. t_2 Decryption time. PSNR after reconstructing the image. ε Redundancy of key.] 420

4.17 Mean square distance between different clustering results of a single image with a unique set of parameters. Values less than 0.05 are written in bold. Images are shown in Figure 4.92. 426
4.18 Comparison of *Welsh et al.*'s colorizing method [375] and the two proposed methods. 443
4.19 PSNR values for *Lena* and *Girl* colorizing tests, using different methods. 445
4.20 Subjective tests of the colorizing tests, all values are out of 100. . . . 445
4.21 Description of the results shown in Figure 4.113 and Figure 4.114. . . 460
4.22 Elapsed time and total distortion results for registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed geometric method. 461
4.23 Total distortion results when registering the images shown in Figure 4.115 with each other and themselves using the proposed geometric method. 463
4.24 Elapsed time for registering images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed gradient–descent enhanced method. 465
4.25 Elapsed time for registering images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed Newton enhanced method. 467
4.26 Total distortion results when registering the images shown in Figure 4.115 with each other and themselves using the proposed Newton enhanced method. 469
4.27 Mismatching value of the registering results shown in Figure 4.126. . 472
4.28 False and true positive results of the best LUTs produced by the color spaces under investigation. 487
4.29 False and true positive results of the best BLUTs produced by the color spaces under investigation. 492
4.30 Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the skin dataset in different color spaces. Part I. 494

4.31 Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the skin dataset in different color spaces. Part II. 494
4.32 Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the control dataset in different color spaces. Part I 495
4.33 Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the control dataset in different color spaces. Part II 495
4.34 Best selection for rejecting one component of the color spaces for skin tone filtering. 496
4.35 Image size, frame count, frame rejection basis (τ), cluster number (c), initialization time (t_0), and frame processing time (t_f) for the image sequences shown in Figure 4.146. 505
4.36 Comparison of the proposed method with the literature. [MS: Multi–spectral, NM: Non Model–based, U: Unsupervised] 521
4.37 Source and reference images corresponding to the destination images shown in Figure 4.159 and Figure 4.160. 524
4.38 Comparison of different recoloring method in terms of realtime application, user contribution, and quality of results. All methods are evaluated when performing on $W \times H$ color images and c pairs of watches. 529
4.39 Numerical details about the results in Figures 4.165, 4.166, and 4.167. [m: Fuzziness, c: Number of swatches. ρ_+, ρ_-: Convolution kernels' sizes. t: Elapsed time.] 531

C.1 Computational cost of converting one pixel to a color space. All values are in terms of flops. 614
C.2 Computational cost of converting one pixel to a color space in its $[0, 255]$ normalized formulation. All values are in terms of flops. . . . 615

C.3 Computational cost of basic operations in terms of flops. When nothing else is restricted, all vectors are d–D, all matrices are $d \times d$, and X and $\tilde{X}$ are sets with n members. I is always a $H \times W$ grayscale image. Number of flops indicated by $\star$ are those for applying the same operation on a bunch of values. [**C**: Crisp, **F**:Fuzzy, **V**: Conventional, **N**: Normalized.] 616

List of Figures

2.1 Sample Image. 7
2.2 Representation of the sample image in RGB color space. 8
2.3 Representation of the sample image in $I_1I_2I_3$ color space. 10
2.4 Representation of the sample image in HSI color space. 12
2.5 Representation of the sample image in HSV color space. 13
2.6 Representation of the sample image in YUV color space. 15
2.7 Representation of the sample image in YIQ color space. 16
2.8 Representation of the sample image in YC_bC_r color space. 17
2.9 Representation of the sample image in $HMMD$ color space. 18
2.10 Representation of the sample image in $CIE-XYZ$ color space. . . . 20
2.11 Representation of the sample image in $CIE-Luv$ color space. 21
2.12 Representation of the sample image in $CIE-Lab$ color space. 22
2.13 Representation of the sample image in $CIE-LHC$ color space. . . . 23
2.14 Values of the function f_{CIE} defined in (2.1.13). 24
2.15 Representation of the sample image in IR_gB_y color space. 25
2.16 Representation of the sample image in $IR_gB_y^+$ color space. 26
2.17 Representation of the sample image in $Nrgb$ color space. 27
2.18 Representation of the sample image in $YC_bC_r^\circ$ color space. 29
2.19 Representation of the sample image in YIQ° color space. 30
2.20 Representation of the sample image in YUV° color space. 31
2.21 Representation of the sample image in HSI^c color space. 32
2.22 Representation of the sample image in HSV^c color space. 33

2.23 Representation of the sample image in RGB^r color space. 35
2.24 Representation of the sample image in TSL color space. 36
2.25 Implementation of the *cief* function. 41
2.26 Implementation of the *colorspaces* function. 42
2.27 Implementation of the *csmax* function. 43
2.28 Implementation of the *csname* function. 43
2.29 Sample results of the *csname* function. 44
2.30 Implementation of the *templatex2y* function. 45
2.31 Implementation of the *rgb2x* function. 47
2.32 Implementation of the *X2rgb* function. 47
2.33 Results of the proposed color transfer methods. (a) and (b) Recoloring. (c) and (d) Colorizing. (a) Original color image. (b) Recolorized image. (c) Original grayscale image. (d) Colorized image. 62
2.34 A typical color transfer system. 64
2.35 Number of scientific publications dedicated to content–based pornography filtering in the years of 1996 to 2004. 74
2.36 Sample result of the proposed recoloring method on still images. (a) Source image. (b) Reference image. (c) Recolored image. 86
2.37 *Bingol*, *Turkey*, (©Digital Globe). (a) Before the earthquake 2002–07–15. (b) After the earthquake 2003–05–02. 89
2.38 Implementation of the *pca* function. 92
2.39 Implementation of the *pcamask* function. 93

3.1 Implementation of the *fpca* function. 100
3.2 Pseudo–code of the proposed general fuzzy clustering algorithm. . . . 107
3.3 Implementation of the *vecangle* function. 109
3.4 Implementation of the *compbv* function. 111
3.5 Implementation of the *decompbv* function. 112
3.6 Implementation of the *makerr* function. 112
3.7 Typical shape of the proposed interpolation kernel, $B^{w,h}_{\tau,N}(x,y)$. 114

3.8 Proposed block–wise interpolation, (a) sample problem and (b) the solution with $SNR > 22dB$. 116
3.9 Implementation of the *interpr* function. 116
3.10 Typical shape of the proposed membership function and α and β parameters. 118
3.11 Log–magnitude of the membership function ($B_{\alpha,\beta}$). (a) $\alpha = 0.9$ with different values of β. (b) $\beta = 0.1$ with different values of α. 119
3.12 Implementation of the *bw* function. 120
3.13 Implementation of the *butterworth* function. 121
3.14 Implementation of the *psnr* function. 125
3.15 Implementation of the *snr* function. 126
3.16 Possible highlighted loci using color spaces. In each case solid lines show the borders of the locus and dashed lines illustrate the skeleton. (a) Planar. (b) Cylindrical. (c) Angular. (d) Possible shape. 131
3.17 An image taken from eight objects with different colors. 136
3.18 (a) A non–homogenous set. (b) Distribution of Euclidean distances. . 139
3.19 (a) Spherical cluster. (b) Elliptical cluster. (c) Cylindrical cluster. . . 140
3.20 Implementation of the *PCAImembership* function. 142
3.21 Implementation of the *PCAImembershipMask* function. 143
3.22 Fuzzification of *Peppers* according to the shown query region shown with different values of β. (a) Original image overlayed the query region, (b) $\beta = 0.1$, (c) $\beta = 0.6$, and (d) $\beta = 0.9$. 144
3.23 Flowchart of the proposed query region extraction. 147
3.24 Implementation of the *SegSim* function. 148
3.25 Implementation of the *SegSimMask* function. 149
3.26 Implementation of the *QuerySegment* function. 150
3.27 Query region extraction results using LPRE. (a) Original image and the query region. (b) Extracted region. 151

3.28 Sample erroneous result of LPRE–based region extraction. (a) Original image and the query region. (b) Extracted region. (c) Color vectors of the query region and the extracted region. 152
3.29 Typical distribution of different distances. (a) LPRE. (b) LLPRE. . . 153
3.30 Typical shape of the function $f(x)$ defined in (3.4.147) with $n = 2$, $\alpha = 5$, and $\beta = 0.001$. 154
3.31 Implementation of the *EHL* function. 154
3.32 Shapes of $f(x)$ defined in (3.4.147) for different values of n with fixed α and β. 155
3.33 Shapes of $f(x)$ defined in (3.4.147) for different values of α with fixed n and β. 156
3.34 Shapes of $f(x)$ defined in (3.4.147) for different values of β with fixed n and α. 156
3.35 Bi–tree decomposition method. 160
3.36 Implementation of the *DoPCASeg* function. 163
3.37 Implementation of the *DoPCASeg12* function. 164
3.38 Implementation of the *DoPCASeg2* function. 165
3.39 Implementation of the *DoPCASeg4* function. 166
3.40 Implementation of the *PCAMerge* function. 167
3.41 Implementation of the *PCASegment* function. 168
3.42 Implementation of the *PUTR* function. 169
3.43 Implementation of the *SetR* function. 169
3.44 Implementation of the *setri* function. 170
3.45 Implementation of the *sets* function. 170
3.46 Implementation of the *merge* function. 171
3.47 Implementation of the *merged* function. 171
3.48 Implementation of the *depthof* function. 172
3.49 Proposed cylindrical clusters. 173
3.50 Implementation of the *FPCAC3d* function. 174

3.51 Implementation of the *FPCAC3Dp* function. 175
3.52 Implementation of the *FFPCAC3Dp* function. 176
3.53 Implementation of the *FPCAC3dre* function. 177
3.54 Implementation of the *LFPCAC3Dp* function. 178
3.55 Proposed cylindrical agent. 180
3.56 Three converged cooperative sets results for the image shown in Figure 4.37–b with $c = 2$, $m = 1.07$, and $\varepsilon = 5$. Top: The corresponding segments. Middle: The clusters. Bottom:The cluster spread histograms. 183
3.57 Three converged cooperative sets results for the image shown in Figure 4.37–a with $c = 2$, $m = 1.07$, and $\varepsilon = 5$. Top: The corresponding segments. Middle: The clusters. Bottom:The cluster spread histograms. 184
3.58 Freely selected samples of three different materials. (a) Skin. (b) Sky. (c) Leaf. 186
3.59 The linear representation of the samples shown in Figure 3.58. (a) Skin. (b) Sky. (c) Leaf. 187
3.60 Principal color of the samples shown in Figure 3.58 extracted by the proposed method. (a) Skin. (b) Sky. (c) Leaf. 187
3.61 Implementation of the *BPCALinSeg* function. 189
3.62 Implementation of the *im2eig* function. 194
3.63 Implementation of the *eig2im* function. 194
3.64 Implementation of the *REmap* function. 195
3.65 Implementation of the *Bim2eig* function. 195
3.66 Implementation of the *Beig2im* function. 196
3.67 Implementation of the *Bdecompose* function. 197
3.68 Flowchart of the proposed color image compression method. 200
3.69 Flowchart of the proposed color image decompression method. 201
3.70 Implementation of the *FPCACim2eig* function. 205
3.71 Implementation of the *FPCACeig2im* function. 206
3.72 Flowchart of the proposed copy protection algorithm. 208

3.73 Implementation of the *FPCACEncrypt* function. 210
3.74 Implementation of the *FPCACDecrypt* function. 211
3.75 Homogenous swatch bank. 212
3.76 Implementation of the *BPCAEncrypt* function. 215
3.77 Implementation of the *BPCACode* function. 215
3.78 Implementation of the *BPCADecrypt* function. 216
3.79 Flowchart of the proposed copy protection method. 217
3.80 Typical input and output of the proposed colorizing method. (a) Source image. (b) Index map. (c) Reference image and the representative swatches. (d) Result. 222
3.81 Sample result of the proposed recoloring method. (a) Source Image and the overlayed swatches. (b) Reference image and the overlayed swatches. (c) Fuzzy membership of source pixels to the flower group. (d) Fuzzy membership of source pixels to the leaf group. (e) Results of the proposed single–swatch recoloring according to the flower swatches. (f) Results of the proposed single–swatch recoloring according to the leaf swatches. (g) Final result. 233
3.82 Implementation of the *recolor* function. 241
3.83 Flowchart of the proposed color transfer method. 242
3.84 Implementation of the *CenterFP* function. 246
3.85 Implementation of the *ErrorFP* function. 246
3.86 Implementation of the *FindRegisteration* function. 247
3.87 Implementation of the *FindRegisterationNewton* function. 247
3.88 Implementation of the *FindRotation* function. 248
3.89 Implementation of the *FindScaling* function. 248
3.90 Implementation of the *FindScalinginternal* function. 249
3.91 Implementation of the *FindTranslation* function. 249
3.92 Implementation of the *merget* function. 250
3.93 Implementation of the *getFP* function. 250

3.94 Implementation of the *ReadFP* function. 251
3.95 Implementation of the *Hessiant* function. 251
3.96 Implementation of the *Jacobiant* function. 252
3.97 Implementation of the *register* function. 252
3.98 Implementation of the *registerfp* function. 253
3.99 Implementation of the *RegisterMerge* function. 253
3.100Implementation of the *SaveFP* function. 254
3.101Implementation of the *showFP* function. 254
3.102Flowchart of the proposed registration method. 258
3.103Pseudo–code of the proposed algorithm for estimating the scaling. . . 263
3.104Pseudo–code of the proposed algorithm for estimating the rotation. . 265
3.105Pseudo–code of the proposed algorithm for estimating the translation. 266
3.106Implementation of the *LUTcensor* function. 278
3.107Implementation of the *chi2* function. 279
3.108Implementation of the *censor* function. 280
3.109Lines corresponding to the skin and non–skin patches. Note the converging pattern of skin–related lines. Refer to the text for more details. 282
3.110Implementation of the *PCAdistance* function. 283
3.111Implementation of the *PcaCensor* function. 284
3.112Implementation of the *singlecd* function. 292
3.113Result of the proposed recoloring process for a single–frame sequence. (a) Source image. (b) Reference image. (c) and (d) Fuzzy membership maps. (e) and (f) Results of single–swatch recoloring. (g) Final result elapsing less than 2 seconds. 295
3.114Implementation of the *FPCAFuzzificate* function. 297
3.115Fuzzification results of the proposed method for two–cluster case with different distances pairs and different values of m. 298
3.116Flowchart of the proposed image sequence recoloring. 300

4.1 Eight homogenous regions in some standard images used for evaluating the proposed dimension reduction method. (a) Nose of *Mandrill*, (b) Wool of *Mandrill*, (c) Lateral part in nose of *Mandrill*, (d) Hair of *Girl*, (e) Skin of *Lena*, (f) Red pepper of *Peppers*, (g) Yellowish–Green pepper of *Peppers*, (h) Green pepper of *Peppers* (images are not proportionally resized). 309
4.2 Results of the proposed dimension reduction process applied on the samples shown in Figure 4.1 (images are not proportionally resized.). 310
4.3 Performance comparison of the distance functions in fuzzifying a given color image according to a query region. (a) Original image adopted from [318] along with the query region, (b) Euclidean. (c) Mahalanobis. (d) LPRE. 318
4.4 Performance comparison of the distance functions in query region extraction. (a) Original image along with the query region, (b) Euclidean. (c) Mahalanobis. (d) LPRE. 320
4.5 Performance comparison of the distance functions in homogeneity decision. (a) Sample image adopted from [318]. (b), (c), and (d) Randomly selected regions sorted in terms of different homogeneity criteria in a descending fashion (b) Euclidean. (c) Mahalanobis. (d) LPRE. 322
4.6 Sample images and query regions. 323
4.7 Ground–truth results of query region extraction for the images shown in Figure 4.6 with corresponding query regions. 325
4.8 Average precision and recall rates of the proposed query region extraction method using LPRE, with varying p. 328
4.9 Average precision and recall rates of the proposed query region extraction method using LPRE, with varying ρ. 328
4.10 Elapsed time of the proposed query region extraction using LPRE, with varying ρ. 329

4.11 (a) Effects of changing p in query region extraction. (a) Effects of changing ρ in query region extraction. 330
4.12 Typical results of the repeatability test. 331
4.13 Typical results of the robustness test. 331
4.14 Samples images and corresponding query regions. 332
4.15 Query region extraction using the LPRE. 333
4.16 Query region extraction using the proposed LLPRE. 334
4.17 Color vectors of the query region, the result of LPRE (shown in Figure 4.15), and the result of LLPRE (shown in Figure 4.16). 335
4.18 Result of the proposed query region extraction for alpha estimation. (a) Original image used as the background adopted from [2]. (b) Foreground image. (c) Canonical method. (d) Proposed method. 338
4.19 Magnified portions of the the montage images shown in Figure 4.18, (a) Canonical method. (b) Proposed method. 339
4.20 Some of the test images, (a)-(c) standard images, (d)-(l) adopted from [318].340
4.21 Sample image and four base regions. 341
4.22 Clusters corresponding to the base regions shown in Figure 4.21. . . . 342
4.23 Shapes of the clusters belonging to the material highlighted in Figure 4.21 in different color spaces. 343
4.24 Results of applying the proposed clustering method on an artificial data. (a) Data. (b) Results. 345
4.25 Results of applying the proposed clustering method on image shown in Figure 4.20–a when using different number of clusters. 346
4.26 Results of applying the proposed clustering method on image shown in Figure 4.20–e when using different number of clusters. 347
4.27 Repeatability test on the image shown in Figure 4.20–b. 348
4.28 Histogram of the correlation of clustering results performed on the same image. 348

4.29 Results of different fuzziness values on the clusters corresponding to the image shown in Figure 4.20–h. 349
4.30 Results of different fuzziness values on the clusters corresponding to the image shown in Figure 4.20–f. 350
4.31 Computational cost and iteration count of the proposed clustering method for different values of m for the image shown in Figure 4.20–i. (a) Iteration count, (b) Elapsed time. 351
4.32 Computational cost and iteration count of the proposed clustering method for different values of c for the image shown in Figure 4.20–i. (a) Iteration count, (b) Elapsed time. 351
4.33 Results of applying the proposed segmentation method on the test images shown in Figure 4.20. 354
4.34 Shapes of the clusters corresponding to the segmentation results shown in figure 4.33, computed by the proposed method. 355
4.35 Results of applying the conventional FCM method on the test images shown in Figure 4.20. 356
4.36 Shapes of the clusters corresponding to the segmentation results shown in figure 4.35, computed by the FCM method. 357
4.37 Some of the test images, (a)–c Standard images, (d)–l Adopted from [318]. 358
4.38 Consecutive iterations of one set of cooperating agents working on the image shown in Figure 4.37–a. Top: the segmentation results. Bottom: clusters. 360
4.39 Results of the proposed clustering method for the image shown in Figure 4.37–a. (a) Decline of $\log_{10} J$ in consecutive iterations. (b) dx values corresponding to each iteration. 360
4.40 Consecutive iterations of one set of cooperating agents working on the image shown in Figure 4.37–g. Top: the segmentation results. Bottom: clusters. 361

4.41 Consecutive iterations of one set of cooperating agents working on the image shown in Figure 4.37–g. Top: the segmentation results. Bottom: clusters. 361
4.42 Results of three different races on the image shown in Figure 4.37–d. 363
4.43 Results of the proposed clustering method performed on the image show in Figure 4.37–e with different values of c. 363
4.44 Results of the proposed clustering method performed on the images shown in Figure 4.37. 365
4.45 Values of $\tilde{\varrho}$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree method and right: proposed bi–tree method. 367
4.46 Values of $\log_4 n$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree. 368
4.47 Values of $\tilde{\varphi}$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree. 369
4.48 Values $\frac{\tilde{\varepsilon}_1}{\varepsilon_1}$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree. 370
4.49 Elapsed time of the decomposition methods for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree. . . . 371
4.50 Sample images. a) *Peppers*. b) *House*. c) *Splash*. a,b,c) Courtesy of USC–SIPI, Signal and Image Processing Institute at the University of Southern California. d) *Food Peppers*, courtesy of *freeimages.co.uk*. e) *Toy*, courtesy of Computational Vision at Caltech, *http://www.vision.caltech.edu*. 372
4.51 Results of three times application the proposed method on the image *Peppers*. Top) Results of segmentation. Bottom) Principal colors. . . 374
4.52 Effects of m on the results of the proposed method on the image *House*. Top) Results of segmentation. Bottom) Principal colors. 375
4.53 Effects of c on the results of the proposed method on the image *Splash*. Top) Results of segmentation. Bottom) Principal colors. 375

4.54 Comparison of the proposed method with the FPCAC described in Section 3.5.2. 377
4.55 Results of the proposed compression method. (a) *Lena*: PSNR=35dB, $\lambda = 3.13$ (b) *Mandrill*:PSNR=31dB, $\lambda = 1.79$ (c) *Girl*: PSNR=35dB, $\lambda = 2.86$ (d) *Airplane*:PSNR=35dB, $\lambda = 3.13$ (e) *Peppers*:PSNR=31dB, $\lambda = 2.50$ (f) *Couple*:PSNR=36dB, $\lambda = 3.13$ 378
4.56 Results of the proposed compression method on standard image *Lena* for different values of ε_1. (a)$PSNR_0$ and $PSNR_+$, (b) Compression ratio 379
4.57 (a) Original image adopted from [318], (b) result of the bi–tree decomposition method, (c) Emap, and (d) Rmap. 381
4.58 (a) Original image , (b) result of the bi–tree decomposition method, (c) Emap, and (d) Rmap. 382
4.59 Eigenimages of the image shown in Figure 4.57–a, (a)pc_1, (b)pc_2, (c)pc_3. 383
4.60 Eigenimages of the image in Figure 4.58–a, (a)pc_1, (b)pc_2, and (c)pc_3. 383
4.61 Histograms of pc_1, pc_2, and pc_3 for the image shown in Figure 4.57–a. 384
4.62 Histograms of pc_1, pc_2, and pc_3 for the image shown in Figure 4.58–a. 385
4.63 Distribution of the energy between the three eigenimages for different values of ε_1 and ϱ. (a)κ_{pc_1}. (b)κ_{pc_2}. (c)κ_{pc_3}. 386
4.64 Results of reconstructing an image from its eigenimages. (a) Original image adopted from [318]. (b) Using all eigenimages (PSNR=60dB). (c) ignoring one eigenimage (PSNR=38dB). (c) ignoring two eigenimages (PSNR=31dB). 387
4.65 Results of reconstructing an image from its eigenimages. (a) Original image. (b) using all eigenimages (PSNR=62dB). (c) ignoring one eigenimage (PSNR=40dB). and (c) ignoring two eigenimages (PSNR=31dB). 388

4.66 PSNR values of image reconstruction using. (a)All three eigenimages. (b)Only two eigenimages. (c)Just one eigenimage for different values of ε_1 and ϱ. 390
4.67 Some of the sample images used for testing the performance of the proposed compression method. 391
4.68 Results of applying the proposed compression method applied on the images shown in Figure 4.67. (for details see Table 4.13). 392
4.69 Exaggerated error of the proposed compression method for the images shown in Figure 4.67. 393
4.70 Results of the proposed watermarking method performed on the image shown in Figure 4.57–a. (a) Watermark data. (b) Watermarked image with PSNR=35dB. (c) Exaggerated difference between the original image and the watermarked image. (d) Extracted watermark data. . 395
4.71 Results of the proposed watermarking method performed on the image shown in Figure 4.58–a. (a)Watermark data. (b)Watermarked image with PSNR=37dB. (c) Exaggerated difference between the original image and the watermarked image. (d) Extracted watermark data. . 396
4.72 Sample watermarks. 397
4.73 Some attacked watermarked images, corresponding items in Table 4.14 are: (a):1, (b):2, (c):3, (d):5, (e):9, (f):11, (g):12, (h):13, (i):14, (j):23, (k):28, and (l):29 399
4.74 Extracted watermarks from attacked images of Figure 4.73. 400
4.75 Some sample images. 402
4.76 Eigenimages extracted from the sample images shown in Figure 4.75 using the proposed method. Dynamic range of the images is exaggerated to give a better visual appearance. (a) First eigenimage. (b) Second eigenimage. (c) Third eigenimage. Number of clusters is set to 5.404
4.77 Elapsed times by the proposed method. (a) Extracting the eigenimages. (b) Reconstructing the image. 405

4.78 Energy compaction of the eigenimages. (a) *Peppers*. (b) *Lena*. (c) *Mandrill*. 406
4.79 PSNR values for different number of clusters when neglecting the third eigenimage. 408
4.80 PSNR values for different number of clusters after watermarking. . . 408
4.81 Results of neglecting the third eigenimage. 409
4.82 Results of the proposed watermarking method. 409
4.83 Watermark signals extracted from images shown in Figure 4.82. . . . 409
4.84 Exaggerated difference between original images shown in Figure 4.72 and watermarked images shown in Figure 4.83. 410
4.85 Results of extracting the watermark using values of $c = 1, \cdots 9$. The actual watermarking is performed using $c = 5$. 411
4.86 Attacked watermarked images. 413
4.87 Extracted watermarks from the attacked watermarked images shown in Figure 4.86. 414
4.88 Some sample images. (a) *Vincent van Gogh*s *Cafe Terrace on the Place du Forum, Arles, at Night*, *Arles*, September 1888, oil on canvas, image from the Vincent van Gogh Gallery, *www.vangoghgallery.com*. (b) and (c) Courtesy of *Oliver Deussen*. (a), (b), and (c) Adopted from [297]. (d) *Mc. Cormic Creck State Park, Indiana* by *Mike Briner*, *mbphoto@spraynet.com*, *www.mikebrinerphoto.com*. (e) *Hanging Lake* by *Brent Reed*, *brent@reedservices.com*. (f) Adopted from [92]. 418
4.89 Encrypted versions of the images shown in Figure 4.88, using proposed algorithm. 420
4.90 Decrypted versions of the images shown in Figure 4.89, using proposed method. 421
4.91 Sequential results of proposed algorithm. (a) Original image, courtesy of Kodak *PhotoCD* (PCD0992). (b) Results of FPCA clustering. (c) Encrypted image. (d) Decrypted image. 423

4.92 Results of applying FPCAC with a unique set of parameters on a single image for 9 different runs. Original image, courtesy of Signal and Image Processing Institute at University of Southern California. 424
4.93 Different encryptions of a single image. 425
4.94 Decryption an image with an invalid key when C and m are known. (a) Original image. (b) and (c) Results of FPCAC. (d) and (e) Decryption with correct keys. (f) and (g) Decryption with invalid (switched) keys. 427
4.95 Sample results of the proposed encryption/decryption method. (a), (d), (g), and (j) Original images. (b), (e), (h), and (k) Encrypted images. (c), (f), (i), and (l) Decrypted images. (a) *Peppers*, courtesy of Signal and Image Processing Institute at the University of Southern California. (d) *House*, courtesy of The Ohio State University. (g) *Skulls*, adopted from *www.artchive.com* 431
4.96 Effect of the choice of N. Image courtesy of *Shohreh Tabatabaii Seifi* and *Ali Qanavati*. 432
4.97 Effect of the choice of A. Image courtesy of *Shohreh Tabatabaii Seifi* and *Ali Qanavati*. 433
4.98 Effect of the choice of q. 434
4.99 Trial to decrypt an image using a key from a different encryption code of its own. (a) Original image. *Pills*, courtesy of *Karel de Gendre*. (b) and (c) Different encryptions. (d) and (e) Decryption using own keys. (f) and (g) Decryption using each other's keys. 435
4.100Adaptivity of JPEG–based compression and the proposed method. a) Original image. b) JPEG2000. c) JPEG with chrominance down–sampling. d) JPEG without chrominance down–sampling. 437
4.101Utilization of the proposed method in the image sequence *Clair*. . . . 438
4.102Segmented version of test images, (a) *Girl*, (b) *Lena*, (c) *Barbara*, (d),(e),(f),(g) adopted from *Welsh et al.* [375], and (h) adopted from *Levin et al.* [238]. 439

4.103A sample from [375] along with the results of the proposed methods. (a) Reference image, (b) Source image. (c) Result of [375]. (d) Result of (3.9.206). (d) Results of (3.9.207). (a),(b),(c) Adopted from [375]. . 440
4.104A sample from [375] along with the results of the proposed methods. (a) Reference image, (b) Source image. (c) Result of [375]. (d) Result of (3.9.206). (d) Results of (3.9.207). (a),(b),(c) Adopted from [375]. . 441
4.105Results of the proposed colorizing methods. (a),(b),(e),(f),(i), and (j) Reference image is *Lena*. (c),(d),(g),(h),(k), and (l) Reference image is *Girl*. (a),(e),(i),(c),(g), and (k) Colorized by (3.9.206). (b),(d),(f),(h),(j), and (l) Colorized by (3.9.207). 444
4.106Comparison of the proposed colorizing methods with the method by *Levin et al.* [238]. (a) Original image adopted from *Levin et al.* [238]. (b) Result of [238]. (c) Result of the method proposed in (3.9.206). (d) Result of the method proposed in (3.9.207). 447
4.107Sample images. (a), (b), (c), (d), (i), and (k) Source images. (e), (f), (g), (h), (j), and (l) Reference images. (a), (b), (e), (f), (i) and (j) Adopted from *Reinhard et. a.* [297]. (c), (g) Adopted from *Greenfield et al.* [156]. (d), (h) Adopted from *Chang et al.* [92]. (k), (l) Adopted from *www.webshots.com* with permission of the authors. (k) *Mc. Cormic Creck State Park, Indiana* by *Mike Briner*, *mbphoto@spraynet.com*, *www.mikebrinerphoto.com*. (l) *Hanging Lake* by *Brent Reed*, *brent@reedservices.com*.449
4.108Results of the proposed recoloring method applied on sample images of Figure 4.107. Source images are shown in Figures 4.107–a, 4.107–b, 4.107–c, 4.107–d, 4.107–i, and 4.107–k, respectively. Reference images are shown in Figures 4.107–e, 4.107–f, 4.107–g, 4.107–h, 4.107–j, and 4.107–k respectively. Used settings are (a) $\beta = 0.95$, $R = \{Sky,Water\}$, (b) $\beta = 0.1$, $R = \{Leaves,Sky\}$, (c) $\beta = 0.8$, $R = \{Skulls,\ Background\}$, (d) $\beta = 0.5$, $R = \{Leaves,\ Sky,\ Earth\}$, (e) $\beta = 0.95$, $R = \{Sky,Building,Pavement\}$. 450

4.109Results of other available methods applied on sample images of Figure 4.107. Source images are shown in Figures 4.107–a, 4.107–b, 4.107–c, 4.107–d, and 4.107–i, respectively. Reference images are shown in Figure 4.107–e, 4.107–f, 4.107–g, 4.107–h, and 4.107–j respectively. (a),(b),(e) Method in [297]. (c) Method in [156]. (d) Method in [92]. . 451
4.110Results of the proposed colorizing method. (a) The reference image. (b) The source image. (c) The result of the proposed recoloring method ($\beta = 0.2$, $R = \{Leaves, Bushes, Bark\}$). 455
4.111Results of the proposed recoloring method. (a) and (b) Original images. (c) Segmentation results of a by the FPCAC. (d) Segmentation of b by the FPCAC. (e) The corresponding pallets (Top: a, Bottom: b). (f) The destination when a is the source and b is the reference. (g) The destination when b is the source and a is the reference. 456
4.112Sample images. (a) and (b) Adopted from [297]. (c) and (d) Adopted from [156]. (e) and (f) Adopted from *www.webshots.com* with permission of the authors. (e) *Mc. Cormic Creck State Park, Indiana* by *Mike Briner*, *mbphoto@spraynet.com*, *www.mikebrinerphoto.com*. (f) *Hanging Lake* by *Brent Reed*, *brent@reedservices.com*. (g) and (h) Adopted from [92]. 457
4.113Results of the proposed method when working on the images shown in Figure 4.112. For details see Table 4.21. 458
4.114Results of other approaches. (a) *Reinhard*, *Ashikhmin*, *Gooch*, and *Shirley* [297]. (b) *Greenfield* and *House* [156]. (c) *Chang*, *Saito* , *Nakajima* [92]. For details see Table 4.21. 459
4.115Different transformations of an image. (a) First image. (b) Translated. (c) Rotated and Translated. (d) Rotated, translated, and balanced scaled. (e) Rotated, translated, and unbalanced scaled. 461
4.116Control points on the image shown in Figure 4.115–a. 462
4.117Panchromatic images of the city of *Bam*. (a) 2003-12-04. (b) 2003-12-29.462

4.118Urban portion of the images shown in Figure 4.117. 463
4.119Results of applying the proposed geometric method when registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a.464
4.120Results of using the gradient–descent enhanced proposed method for registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a. 466
4.121Results of using the proposed method for registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed Newton enhanced method. 468
4.122Result of applying the proposed Newton enhanced registration method on the images shown in figure 4.117. 469
4.123Result of applying the proposed Newton enhanced registration method on the images shown in figure 4.118. 470
4.124(a) Sample image. (b) Sample image undergoing scaling, rotation, translation, and a little skewness. 471
4.125Three possible distribution patterns of control points. (a) Spread. (b) Linear. (c) Zonal. 472
4.126Results of registering the images shown in Figure 4.125 on each other using the proposed Newton enhanced method. (a) Spread. (b) Linear. (c) Zonal. 473
4.127Statistics of the images in the erotic dataset. a) Classification of the images into three categories of not manipulated, tagged, and edited. b) Histogram of the width and the height of the images. c) Histogram of the aspect ratios. d) Histogram of bpp of the images. e) Number of people present in the images. f) Is there a face in the image? g) Separation of the images according to the visible sexual organs. h) Categorization in terms of being outdoor or indoor. 478
4.128Skin Dataset. 480
4.129Control Dataset. 480

4.130ROC curves of the OLUT–based skin detector. 481
4.131ROC curves of the BLUT–based skin detection. 482
4.132Comparison of the original color spaces with their manipulated versions in terms the results of the best combination for LUT–based Skin Detection. 484
4.133Best ROC curves for OLUT–based skin detection obtained from the investigated color spaces. 485
4.134Best LUTs produced by the color spaces under investigation. 486
4.135Results of computing the skin map using the best available LUTs. . . 488
4.136Comparison of the original color spaces with their manipulated versions in terms the results of the best combination for BLUT–based Skin Detection. 490
4.137Best ROC curves for BLUT skin detector obtained from the investigated color spaces. 491
4.138Best BLUTs produced by the color spaces under investigation. 491
4.139Results of computing the skin map using the best available BLUTs. . 493
4.140Histogram of the expectation and standard deviation of the sample images in the selected color components. (a),(d) $CIE - a$. (b),(e) $HSV.H$. (c),(f) I_3. (a),(b),(c) Control dataset. (d),(e),(f) Skin dataset. 498
4.141Results of classifying the samples in the two datasets of skin, ((I),(II)), and control, ((III),(IV)), into two classes of skin, ((I),(III)), and non–skin, ((II),(IV)). The top row shows the skin signature. The operation is performed in the three color components of (a) $CIE-a$, (b) $HSV.H$, and (c) I_3. 500
4.142ROC curve of the proposed skin detection method. 500
4.143Principal skin color computed by the proposed method. 501
4.144Sample images. 502
4.145Results of applying the proposed skin detection method on the images shown in Figure 4.144. 503

4.146Some test image sequences (see Table 4.35 for details). 506
4.147Results of applying the proposed segmentation and change detection method on *Claire* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask. 507
4.148Results of applying the proposed segmentation and change detection method on *Flower Garden* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask. 508
4.149Results of applying the proposed segmentation and change detection method on *Foreman* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask. 509
4.150Results of applying the proposed segmentation and change detection method on *Car Phone* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask. 510
4.151Results of applying the proposed segmentation and change detection method on *Miss America* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask. 511
4.152Results of applying the proposed segmentation and change detection method on *Table Tennis* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask. 512
4.153Results of applying the proposed segmentation and change detection method on *Baseball* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask. 514
4.154Multi–spectral images of the city of *Bam*. (a) 2003-12-04. (b) 2003-12-29.516
4.155Three query regions in the image shown in Figure 4.154–a. 517
4.156Results of applying maximum likelihood on the query regions shown in Figure 4.155. (a) Euclidean. (b)Mahalanobis. (c) LPRE. 518
4.157Result of the proposed segmentation method applied on the image shown in Figure 4.154–a. 519

4.158Result of the proposed multi–spectral fuzzy change detection method applied on the images shown in Figure 4.154. 520
4.159Results of the proposed color transfer method. See Table 4.37 for details.522
4.160Results of other available recoloring methods. (a),(b),(e) Method in [297]. (c) Method in [156], (d) Method in [92]. For details see Table 4.37 . 523
4.161Utilization time of different methods in a 1Gflop processor. 526
4.162Proposed four–processor architecture used for the realtime application of the recoloring methods. 528
4.163Simulation of the proposed method in the proposed architecture. (a) Source image. (b) Reference image. (c) Recolored image. 529
4.164Utilization time of different methods in the proposed architecture. . . 530
4.165Results of applying the proposed recoloring method on the *Table Tennis* standard image sequence (for details see Table 4.39). 532
4.166Results of applying the proposed recoloring method on the *Clair* standard image sequence (for details see Table 4.39). 533
4.167Results of applying the proposed recoloring method on the *Flower Garden* standard image sequence. (for details see Table 4.39). 534
4.168Swatches used for recoloring processes shown in Figures 4.165, 4.166, and 4.167. For details see Table 4.39. 535

A.1 Implementation of the *RCIPIS* function. 539
A.2 Implementation of the *LCIPIS* function. 539
A.3 Implementation of the *epsifyfig* function. 540
A.4 Implementation of the *epsifyI* function. 540
A.5 Implementation of the *epsifyf* function. 541
A.6 Implementation of the *grect* function. 542
A.7 Implementation of the *hist3* function. 542
A.8 Implementation of the *I2RLE* function. 543
A.9 Implementation of the *RLE2I* function. 543
A.10 Implementation of the *I2V* function. 544

A.11 Implementation of the *V2I* function. 545
A.12 Implementation of the *splitrgb* function. 545
A.13 Implementation of the *mergergb* function. 546
A.14 Implementation of the *ICor* function. 546
A.15 Implementation of the *jpeg* function. 547
A.16 Implementation of the *pcx* function. 547
A.17 Implementation of the *pdf1d* function. 548
A.18 Implementation of the *pdf2d* function. 549
A.19 Implementation of the *pdf3d* function. 549
A.20 Implementation of the *pdfnd* function. 550
A.21 Implementation of the *normalize* function. 551
A.22 Implementation of the *normalize255* function. 551
A.23 Implementation of the *scale255* function. 552
A.24 Implementation of the *rescale255* function. 552
A.25 Implementation of the *percentile* function. 553
A.26 Implementation of the *themost* function. 554
A.27 Implementation of the *Energy* function. 554
A.28 Implementation of the *ExtractBitPlanes* function. 555
A.29 Implementation of the *ishow* function. 555
A.30 Implementation of the *ifshow* function. 556
A.31 Implementation of the *ShowLayer* function. 557
A.32 Implementation of the *showlayers* function. 557
A.33 Implementation of the *ShowSegmentation* function. 558
A.34 Implementation of the *Showqrs* function. 558
A.35 Implementation of the *showr* function. 559
A.36 Implementation of the *showrb* function. 559
A.37 Implementation of the *showrc* function. 560
A.38 Implementation of the *ShowRGUI* function. 560
A.39 Implementation of the *ShowRORDR* function. 561

A.40 Implementation of the *ShowComposition* function. 561
A.41 Implementation of the *segresult* function. 562
A.42 Implementation of the *hist3d* function. 562
A.43 Implementation of the *showcluster* function. 563
A.44 Implementation of the *ShowColor* function. 563
A.45 Implementation of the *ShowColors* function. 564
A.46 Implementation of the *LaplacianWeight* function. 565
A.47 Implementation of the *RCEput* function. 566
A.48 Implementation of the *RCEWput* function. 566
A.49 Implementation of the *RCFPCAPallet* function. 567
A.50 Implementation of the *RCPallet* function. 567
A.51 Implementation of the *RCPCAev* function. 568
A.52 Implementation of the *RCWPallet* function. 568
A.53 Implementation of the *RCPCAPallet* function. 569
A.54 Implementation of the *RCPCAput* function. 569
A.55 Implementation of the *ShowPallet* function. 570
A.56 Implementation of the *RCput* function. 570
A.57 Implementation of the *ShowPallet3D* function. 571
A.58 Implementation of the *RCSegment* function. 571
A.59 Implementation of the *ShowPalletTree* function. 572

D.1 Implementation of the *MSCPbest* function. 625
D.2 Implementation of the *MSCPvalue* function. 625
D.3 Implementation of the *MultiSegment* function. 626
D.4 (a) *Lena* image. (b) PDF of (a). (c) Values of the function f defined in (D.2.13). (d) Single–layer segmentation result (PSNR=19.8dB). . . 627
D.5 (a) PSNR of a single–layer segmentation of *Peppers* image with varying λ and r. (b) Logarithm of the computational cost for different values of λ and r. 628
D.6 Results of applying the proposed method on a 1–D signal. 630

D.7 Results of applying the proposed method on *unr730* image. (a) Original image adopted from *http://ipagwww.med.yale.edu/∼staib*. (b) Single–layer representation. (c) Two–layer representation. (d) Three–layer representation. (e) Four–layer representation. (f) Five–layer representation. 631
D.8 PSNR versus layer number for *unr730* image. 632

Abstract

It is known since 1988 that utilizing linear dimension reduction gives appropriate lower dimensional representation of homogenous swatches of color images of the nature. Though, it is common to see serious research projects, even dated 2005, which are based on the fixed color space paradigm. In this thesis, first experimental evaluations are performed to compare the principal component analysis (PCA)–based representation of color images of the nature with respective color space–based representations. According to the promising results of the experiments and the theoretical anticipation of the properness of local PCA–based approach, we turn into applications. The PCA model for the color vectors of homogenous swatches defines one principal direction and two less important ones. This framework is utilized for designing outperforming color image processing algorithms such as colorizing, recoloring, compression, segmentation, watermarking, deliberate distortion, and so on. For every algorithm numerous experiments are performed and the results are discussed. Also, wherever possible, the proposed methods are compared to the literature. All algorithms are incorporated into a unique color image processing toolbox for MATLAB.

Acknowledgements

I would like to thank Dr. *Shohreh Kasaei*, my supervisor, for her many suggestions and constant support during this research. She helped me in many hard situations and spent her invaluable time for guiding me through the months of chaos and confusion. I owe her, not only because of giving me the passion for research, but also because of her continuous real–life guidance. Also, I like to appreciate Dr *Amir Daneshgar*, my advisor, who helped me to adopt this thesis to the standards of the mathematics faculty, Sharif University of Technology. I wish to thank heads of the faculty for their invaluable helps during the time I spent there. I thank Dr. *Ebadollah Mahmoudian*, the former dean, Dr. *Zanganeh* and Dr. *Asaraii*.

I have benefited from respective professors inside and outside Sharif University of Technology during my research. Dr. *Saeed Bagheri Shouraki* introduced me with the fuzzy theory. I thank Dr. *Mehrdad Mirshams Shahshahani* for letting me to participate in the IPM international workshop on computer vision and the short course on SVM. I would like to appreciate the valuable discussions and suggestions made by professor *M. Nakamura* and professor *Y. Kosugi* from Tokyo Institute of technology. I had the opportunity to have useful discussions with *Patrizio Frosini* from the University of Bologna, *Taraneh Eghlidos* from Sharif University of Technology, *Edmond Zahedi* from the National University of Malaysia, and *Jan-Mark Geusebroek* from the University of Amsterdam. Also, I am grateful of these people for supplying me with the papers I needed: *Terry Caelli* from the University of Alberta, *David Gering* (MD) from Massachusetts institute of technology, *Horia F. Pop* from Babes–Bolyai University, *James C Bezdek* from the University of West Florida, *Matthew Turk* from

the University of California Santa Barbara, *Youngha Chang* from Tokyo Institute of Technology, *Mohan Kankanhalli* and *Yan Weiqi* from the National University of Singapore, *Markus Hofmann* from Bell Laboratories, *Louise Ditmore*, *Henry Schneiderman* and *Takeo Kanade* from Carnegie Mellon University, *Klaus Mueller* from Stony Brook University, *Hiroaki Kotera* from Chiba University, *Greg Mori* from Berkeley University of California, *Elad Michael* from Stanford University, *Daniel Cremers* from UCLA, and *Athinodoros Georghiades* from Yale University. I wish to appreciate Professor *Mohammad D. Jafari* for his several interesting lectures in Iran.

During the research on pornography filtering, I was partially supported by *Tebian* institute (*www.tebian.net*). I wish to thank Dr. *Alireza Talebpour* (the head of *Tebian*) from *Shahid Beheshti* university, Iran, and professor *Maria Petrou* from *Surrey* university, Guildford, UK, for their constructive comments and good discussions. Also, I appreciate invaluable aids of *Vahid Vaziri Hamaneh*.

The application of many samples in this thesis was possible only because I received top–rating databases and samples from these people: *Simon Baker* from Carnegie Mellon University, *Aleix M. Martinez* from Ohio State University, *Sami Huovinen* and *Matti Pietikainen* from the University of Oulu, *Michael J. Lyons* from ATR Media Information Science Labs, University of Oulu , and *Linda Diane Decker* from the National Institute of Standards and Technology. Also, I thank *Steven J. Wolf* from the California State University, *Mike Briner*, *Brent Reed*, *Richard Seaman Shohreh Tabataei Seifi*, and *Ali Qanavati* for letting me to use their copyrighted images. I also wish to thank the *Iranian Remote Sensing Center* (IRSC) and *Digital Globe* for providing me with the remote sensing images used in this thesis. Finally, I wish to thank *Robert Tibljas* for his brilliant open–source code for media conversion.

Of course, I am grateful to my family for their patience and love. Without them, this work would never have come into existence. Specially, I wish to thank my father and my mother for backing me up in many problems. I wish to acknowledge the friendship and support of my family members: my father and my mother, *Anahita*, *Azadeh*, *Alaleh*, and *Aida* (my sisters), my father– and my mother–in–law, *Elham*, *Elaheh*, and

Mohammad Mehdi (my bother– and sister–in–laws) and *Vahid* (my brother–in–law).

Finally, I wish to thank the followings: *Kourosh Aliany*, *Masoud Mazloom*, *Bardai Mohabbati*, *Nazila Hafezi*, *Mehdi Mortezaii*, and *Marzieh Niroumand*, for the good time I spent with them.

After all, I owe everything to my beloved wife *Azadeh Yadollahi* for her support and patience.

Karaj, Iran
Arash Abadpour
25 June 2005, 4 Tir 1384.

Chapter 1

Introduction

Image processing has received enormous focus in the last century. This is mainly related to the wide area of applications of image processing. The other important feature of image processing tasks is their triviality when we compare them with what our visual system performs. Hence, a beginner in the field will not get the point of all the mathematics devoted to establish an integrated algorithm for an enormously easy task.

The first image processing research projects were devoted to processing binary and then grayscale images. This was the clear choice at the time, because time had to be spent on developing the first color recording camera. Though, decades passed before digital cameras of acceptable quality and resolution came into market. Frankly speaking, working on color image processing was presumed to be unnecessary even in the last decade.

When researchers started to think about color image processing, the first algorithms were just parallelized versions of the available grayscale algorithms. At that time, it was just common to think of a RGB color image as a set of three parallel grayscale images. Unfortunately, this miss–understanding is extensively common in today's image processing community.

When working with color images, one has to face the correlation between color components and the distribution of energy among them. This two–fold phenomena is both a threat and an opportunity. Thanks to the physical evidences, principal component analysis has proved to be an appropriate tool for both decorrelating color components and compacting the energy. As such, it is the time to quit working with fixed color transformations and face the data–adaptive transformation of PCA.

We will gather experiments to show the appropriateness of PCA for processing color images. Equipped with these experimental results, which are not yet physical proofs, series of color image processing algorithms are developed that use PCA as the basis for transforming color vectors into a more applicable framework. The results demonstrate that this approach is both effective and efficient.

In thesis we work on the application of PCA to color image processing tasks of spotting, query region extraction, segmentation, compression, watermarking, direct copy protection, color transfer (colorizing and recoloring), content based pornography filtering, video segmentation and change detection, and video recoloring. First, the

cylindrical model for homogenous color clusters is used for performing spotting and extracting the region similar to a query. Also, incorporating fuzzy clustering, efficient segmentation methods are proposed. Using the energy compaction embedded in principle component analysis and its local version, algorithms for color image compression and watermarking are developed. Then we use the cylindrical model of homogenous color clusters for transferring color from a color image to a grayscale (colorizing) or a color (recoloring) one. This scheme is also used for reversible deliberate distortion of color images for the sake of copy protection. Using the principal color concept, a serious alternative for look–up table based approach for finding skin in color images of unknown sources is proposed. The thesis contains precise comparison of the proposed algorithm with the available ones. At the end, we examine color image sequences and develop the PCA–based approach for segmentation, change detection, and recoloring of video resources. Everywhere, the PCA–based approach is compared to many fixed color transforms and the results are analyzed.

The main reason for selecting PCA over canonical color transforms is its adaptivity to the ongoing set of vectors. We expect a transform to give uncorrelated vectors with appropriate compaction of energy. When we talk about PCA, the first question is about its computational cost. Although, applying PCA in the spatial domain is massively expensive, there is a handy way to perform PCA transform in the spectral domain, which is comparable in cost to the canonical color transforms. It should be

emphasized that everywhere in this thesis, PCA is applied on homogenous sets of color vectors (doing otherwise is non–meaningful). Chapter 2 gives a brief review of the available literature for some common image processing tasks. Then, Chapter 3 introduces the proposed method. Finally, Chapter 4 holds the experimental results.

Chapter 2

Literature Review

In many applications of vision, color is an important cue (because it is robust against changes in orientation and scaling and can well tolerate occlusion). For example, the *RoboCup* games are held in a field officially defined as "*a square with green carpets and white walls in which two teams of four or five completely black robots are trying to kick a red ball toward two goals colored in blue and yellow, respectively*" [269]. In this atmosphere, vision is the essential tool to recognize objects and is based on the color diversity.

For a soccer player robot, going towards the ball at about $2m/s$ velocity, processing 16 frames per second results in about $62.5mm$ error in each frame which is a real shortcoming. This proves the need for an enough accurate algorithm that is performed fast enough. In this chapter we will briefly review the available literature of color image processing.

Color is the way the *human visual system* (HVS) perceives a part of the electromagnetic spectrum occupying the band between $380nm$ and $780nm$. As the main purpose of this thesis is to develop practical tools for robust processing in color fields, we do not enter the psychological aspects of color perception. An interested reader is referred to the comprehensive discussions in [189, 64, 107].

2.1 Color Spaces

A color space is a method to code a visible waveform. In this section the available color spaces are discussed. First Section 2.1.1 introduces the color spaces and briefly reviews their history. Then, Section 2.1.2 talks about the implementation of color spaces in **CIP**.

2.1.1 Formulations

Although, due to practical reasons, RGB color space is widely used in science and technology, it is not the best choice. For example, when dealing with the images of the nature, RGB suffers from high correlation between its components: 0.78 for r_{BR}, 0.98 for r_{RG} and 0.94 for r_{GB} [162]. Here, r_{xy} denotes the correlation between the two random variables x and y. Also, the RGB color space is proved to be psychologically not intuitive [4] in the way that the human observer has problems imagining pure colors red, green and blue as defined in RGB. Also, RGB is perceptually non–uniform [4, 348] because the correlation between the perceived difference of two colors

Figure 2.1: Sample Image.

and the regarding Euclidian distance in the RGB space is too low.

In the years passed from the birth of color imaging, many different color spaces are proposed and investigated in the color image processing community. These color spaces could be informally categorized into three major categories of HVS–based (including RGB, opponent, and phenomenal color spaces), application specific, and CIE color spaces [277]. Here, we review the most frequently used color spaces. All these color spaces are supported in **CIP**. Figure 2.1 shows the sample image used for comparing different color spaces. For visualizing each color space, after computing the representation of the image shown in Figure 2.1, the new components are assumed as R,G, and B channels and the resulting image is visualized. Figure 2.1 shows the sample image in the RGB color space.

In the late 19–th century, *Ewald Hering* proposed the opponent color theory [277].

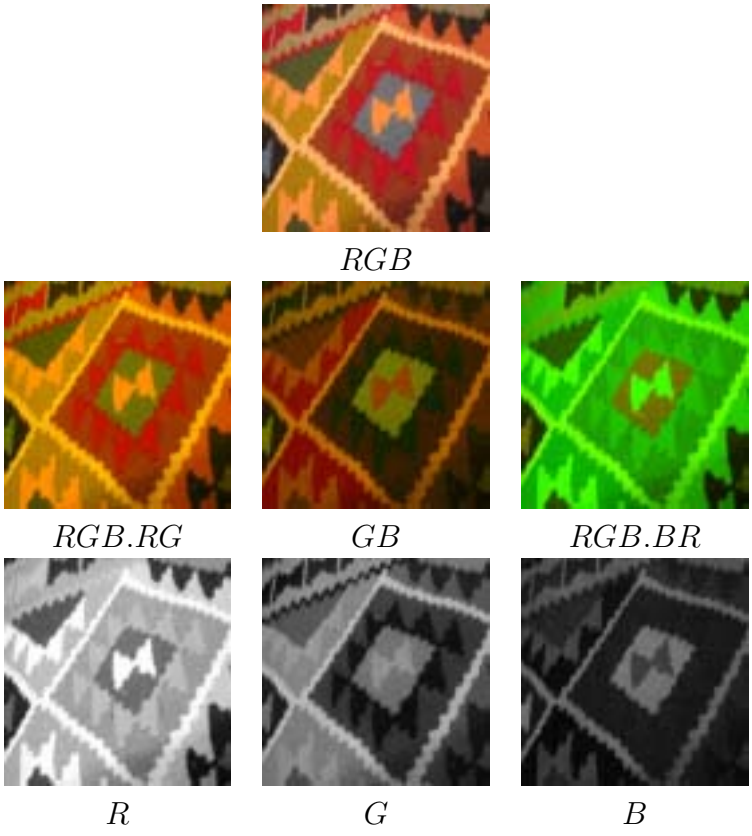


Figure 2.2: Representation of the sample image in RGB color space.

The relating color space is modelled by different researchers like *Judd*, *Adams*, *Hurvich*, *Jamson* and *Guth* [277]. Another work is an excellent color space proposed by *Ohta* as a very good approximation of the Karhunen–Loeve transformation of the decorrelated RGB space (also called $I_1I_2I_3$) [273]. Figure 2.3 shows the sample image represented in this color space.

$$I_1I_2I_3 : \begin{cases} I_1 = \frac{1}{3}(R+G+B) \\ I_2 = \frac{1}{2}(R-B) \\ I_3 = \frac{1}{4}(2G-R-B) \end{cases} . \tag{2.1.1}$$

Phenomenal color spaces, using attributes of the hue and saturation (based on Newton's color circle) have proved to be a natural way to describe human sense of color [4, 165]. Many different color models of this category exist in the literature; like HSI (2.1.2) [350] and HSV (2.1.3) [138, 233]. Although, the phenomenal color spaces are very intuitive, they inherit the device–dependent tendency from the mother space RGB. Also, phenomenal color spaces suffer from the hue discontinuity around 2π and the main shortcoming of non–uniform perception. Figures 2.4 and 2.5 show the sample image in HSI and HSV representations, respectively. Several interesting properties of Hue are noted in [327]: it is invariant to highlights at white light sources, and also, for matte surfaces, to ambient light and surface orientation relative to the light source. However, *Poynton* points out several undesirable features of these color spaces, including hue discontinuities and the computation of brightness (lightness,

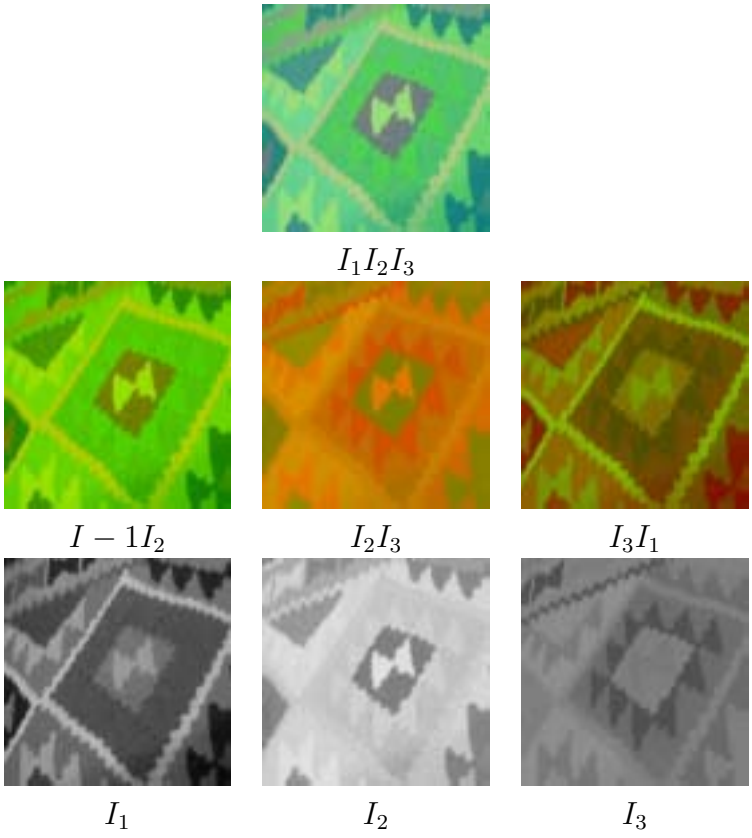


Figure 2.3: Representation of the sample image in $I_1I_2I_3$ color space.

value), which conflicts badly with the properties of color vision [289].

$$HSI: \begin{cases} H = cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \\ S = 1 - \frac{min(R,G,B)}{I} \\ I = \frac{1}{3}(R+G+B) \end{cases}, \tag{2.1.2}$$

$$HSV: \begin{cases} H = \begin{cases} h, B \leq G \\ 2\pi - h, B > G \end{cases} \\ h = cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \\ S = \frac{max(R,G,B)-min(R,G,B)}{max(R,G,B)} \\ V = max(R,G,B) \end{cases}. \tag{2.1.3}$$

Application specific color spaces are those invented for special commercial purposes including the spaces used in printing systems (CMYK (2.1.4) [340]), television systems (YUV (2.1.5) [328], YIQ (2.1.6) [58], YC_bC_r (2.1.7) [5]) and photographic systems (YCC). These color spaces are quite non–intuitive and perceptually non–uniform. Figures 2.6, 2.7, and 2.8 show the sample image represented in YUV, YIQ, and YC_bC_r color spaces, respectively. Also, the MPEG–7 standard defines a new color space called $HMMD$ [251]. Figure 2.9 shows the sample image represented in this color space.

$$CMYK: \begin{cases} C = \frac{\tilde{C}-\tilde{K}}{1-\tilde{K}} \\ M = \frac{\tilde{M}-\tilde{K}}{1-\tilde{K}} \\ Y = \frac{\tilde{Y}-\tilde{K}}{1-\tilde{K}} \\ K = min(\tilde{C},\tilde{M},\tilde{Y}) \\ \tilde{C} = 1-R, \tilde{M} = 1-G, \tilde{Y} = 1-B \end{cases}, \tag{2.1.4}$$

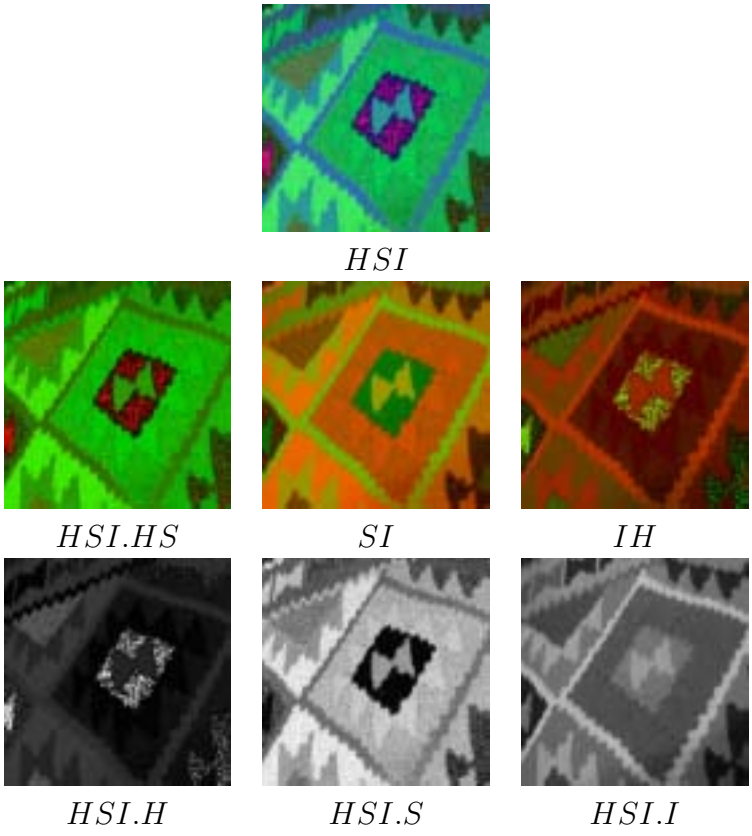


Figure 2.4: Representation of the sample image in HSI color space.

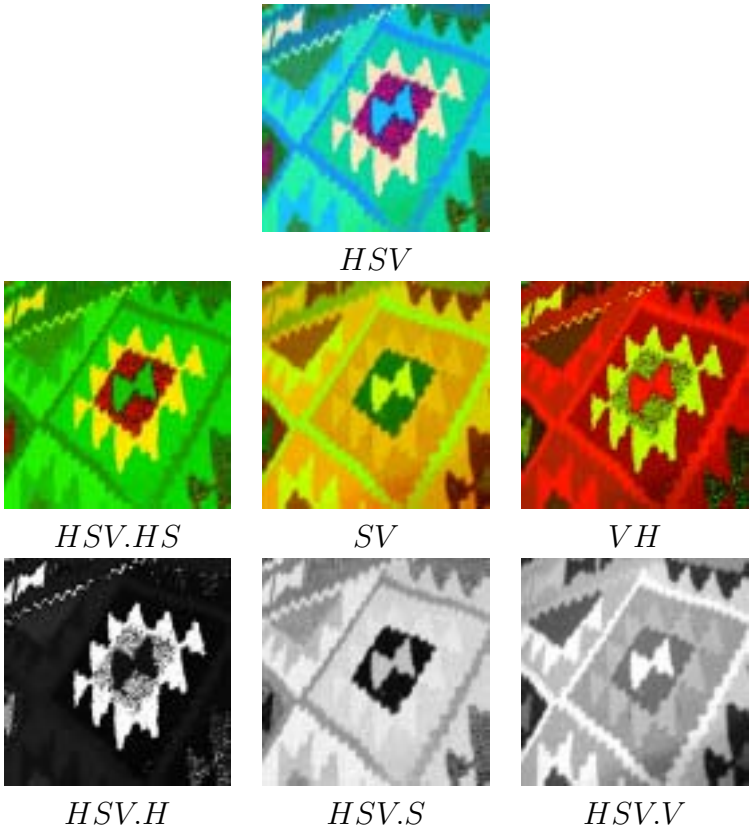


Figure 2.5: Representation of the sample image in HSV color space.

$$YUV : \begin{cases} Y = 0.30R + 0.59G + 0.11B \\ U = -0.15R - 0.29G + 0.44B \\ V = 0.62R - 0.52G - 0.10B \end{cases} , \tag{2.1.5}$$

$$YIQ : \begin{cases} Y = 0.30R + 0.59G + 0.11B \\ I = 0.60R - 0.28G - 0.32B \\ Q = 0.21R - 0.52G + 0.31B \end{cases} , \tag{2.1.6}$$

$$YC_bC_r : \begin{cases} Y = 0.30R + 0.59G + 0.11B \\ C_b = 0.56(B - Y) \\ C_r = 0.71(R - Y) \end{cases} . \tag{2.1.7}$$

$$HMMD : \begin{cases} H = cos^{-1}\Big(\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\Big) \\ Min = min\{R, G, B\} \\ Max = max\{R, G, B\} \\ D = Max - Min \end{cases} , \tag{2.1.8}$$

while only two of Min, Max, and D are sufficient to define a single color realization (here, we use Min and Max)

In 1931, CIE laid down the CIE–1931 standard to make a resolution for the device–dependent tendency of the RGB color space and other spaces based on it. The standard leads to the $CIE - XYZ$ color space as a descriptor of the average human observer (2.1.9). Then, in 1976, CIE introduced two other color spaces named officially as $CIE-Luv$ (2.1.10) and $CIE-Lab$ (2.1.11). The main purpose of defining

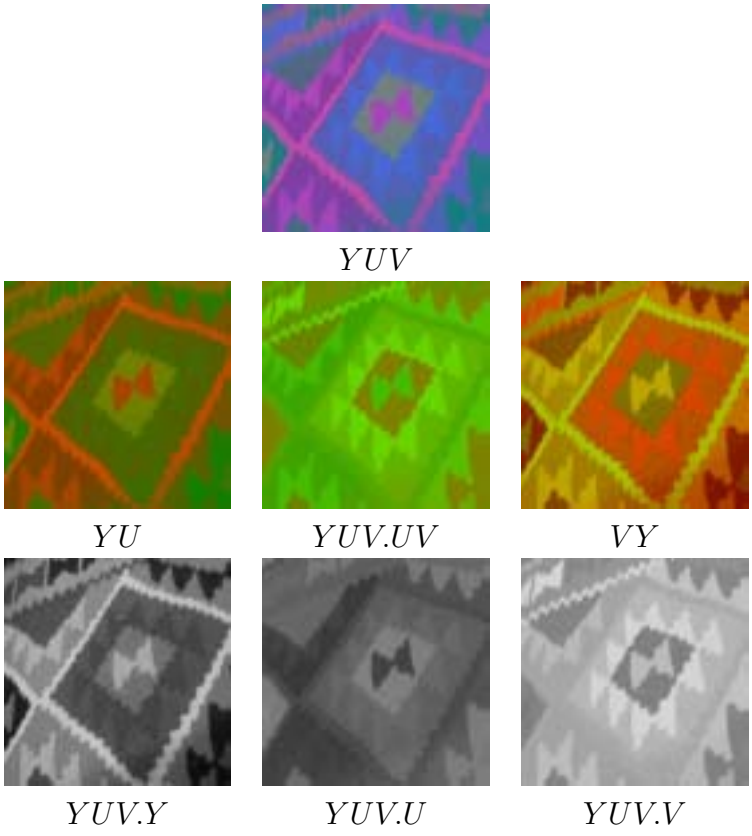


Figure 2.6: Representation of the sample image in YUV color space.

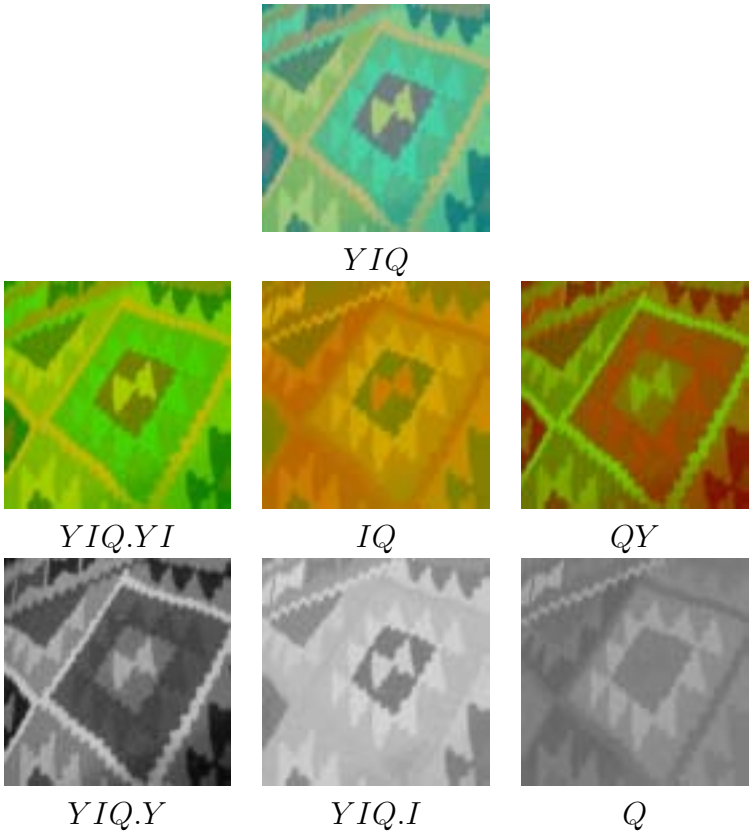


Figure 2.7: Representation of the sample image in YIQ color space.

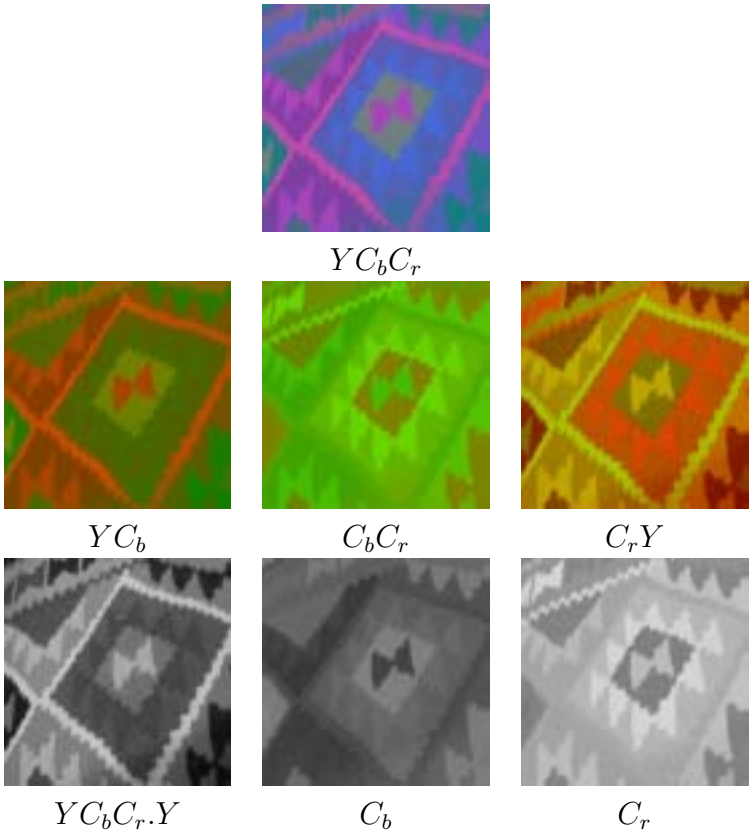


Figure 2.8: Representation of the sample image in YC_bC_r color space.

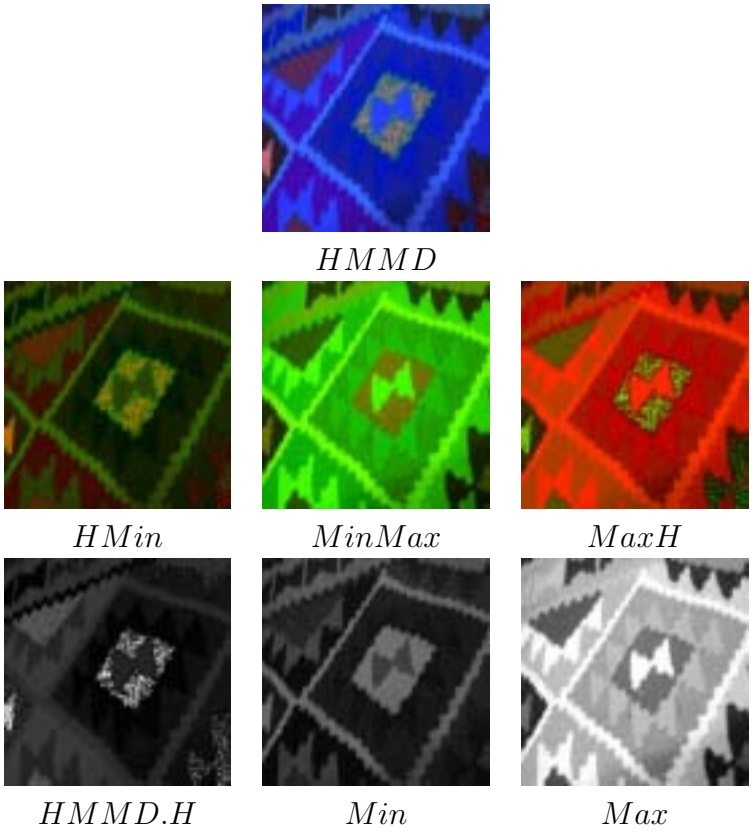


Figure 2.9: Representation of the sample image in $HMMD$ color space.

these color spaces is to provide a perceptually uniform space. Later, it was proved that $CIE-Luv$ is not entirely uniform [239]. After that, $CIE-LHC$ (2.1.12) was defined as the polar version of $CIE-Lab$ [340]. Figures 2.10, 2.11, 2.12, and 2.13 show the sample image represented in $CIE-XYZ$, $CIE-Lab$, $CIE-Luv$, and $CIE-LHC$ color spaces, respectively.

$$CIE-XYZ\begin{cases} X = 0.61R + 0.17G + 0.20B \\ Y = 0.30R + 0.59G + 0.11B \\ Z = 0.00R + 0.07G + 1.12B \end{cases}, \tag{2.1.9}$$

$$CIE-Luv\begin{cases} L = 116 f_{CIE}(\frac{Y}{Y_0}) \\ u = 13L(\acute{u} - \acute{u}_{White}) \\ v = 13L(\acute{v} - \acute{v}_{White}) \\ \acute{u} = \frac{4X}{X+15Y+3Z} \\ \acute{v} = \frac{9Y}{X+15Y+3Z} \end{cases}, \tag{2.1.10}$$

$$CIE-Lab\begin{cases} L = 116 f_{CIE}(\frac{Y}{Y_0}) \\ a = 500(f_{CIE}(\frac{X}{X_0}) - f_{CIE}(\frac{Y}{Y_0})) \\ b = 500(f_{CIE}(\frac{Y}{Y_0}) - f_{CIE}(\frac{Z}{Z_0})) \end{cases}, \tag{2.1.11}$$

$$CIE-LHC\begin{cases} L = 116 f_{CIE}(\frac{Y}{Y_0}) \\ H = tan^{-1}\left(\frac{b}{a}\right) \\ C = \sqrt{a^2+b^2} \end{cases}. \tag{2.1.12}$$

The function $f_{CIE}(x)$ in (2.1.10), (2.1.11), and (2.1.12) is the CIE function:

$$f_{CIE}(x) = \begin{cases} x^{\frac{1}{3}}, x > 0.008856 \\ 7.787x + \frac{16}{116}, otherwise \end{cases}. \tag{2.1.13}$$

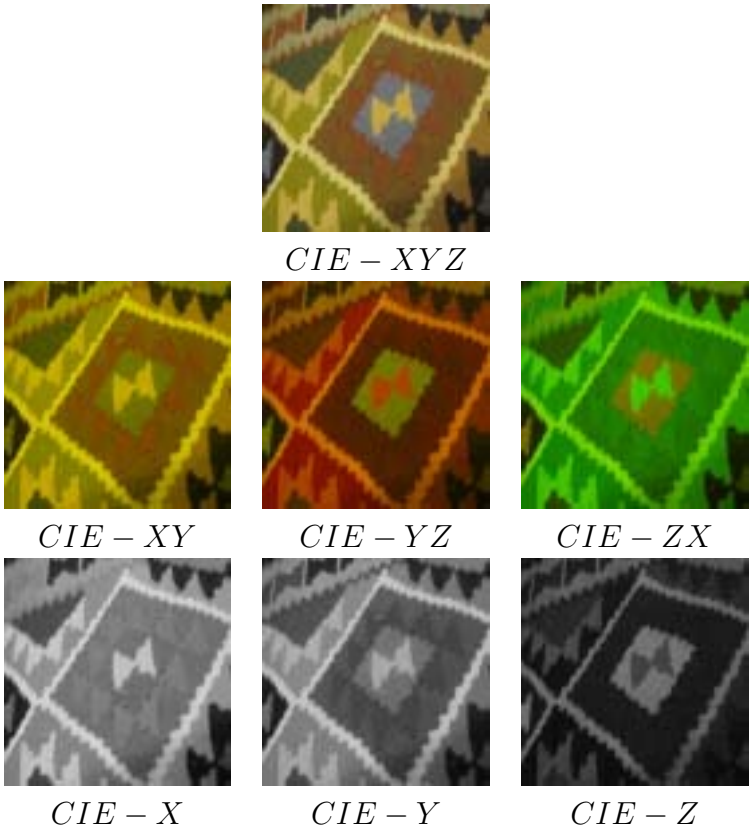


Figure 2.10: Representation of the sample image in $CIE - XYZ$ color space.

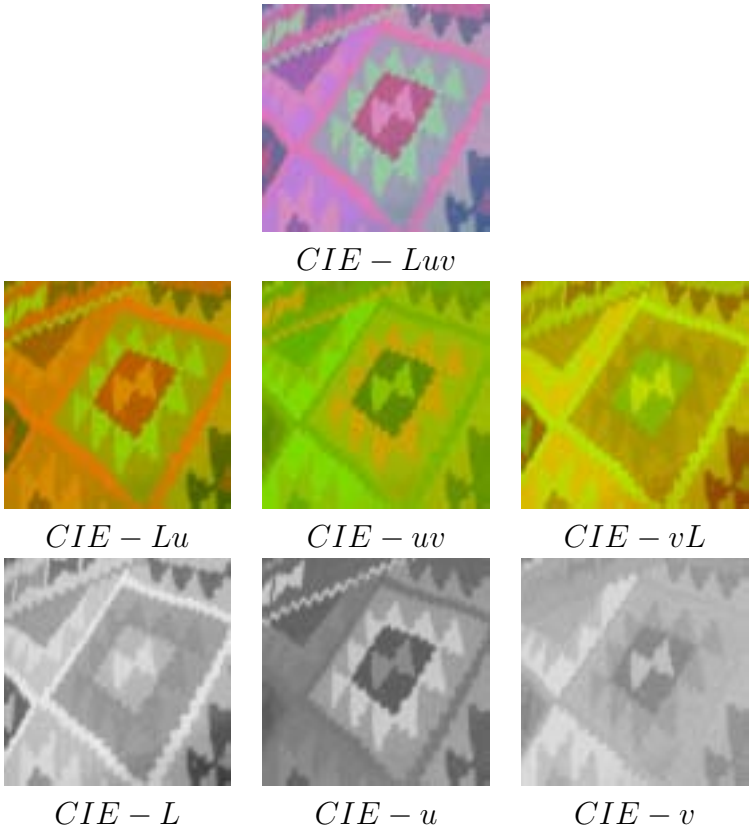


Figure 2.11: Representation of the sample image in $CIE - Luv$ color space.

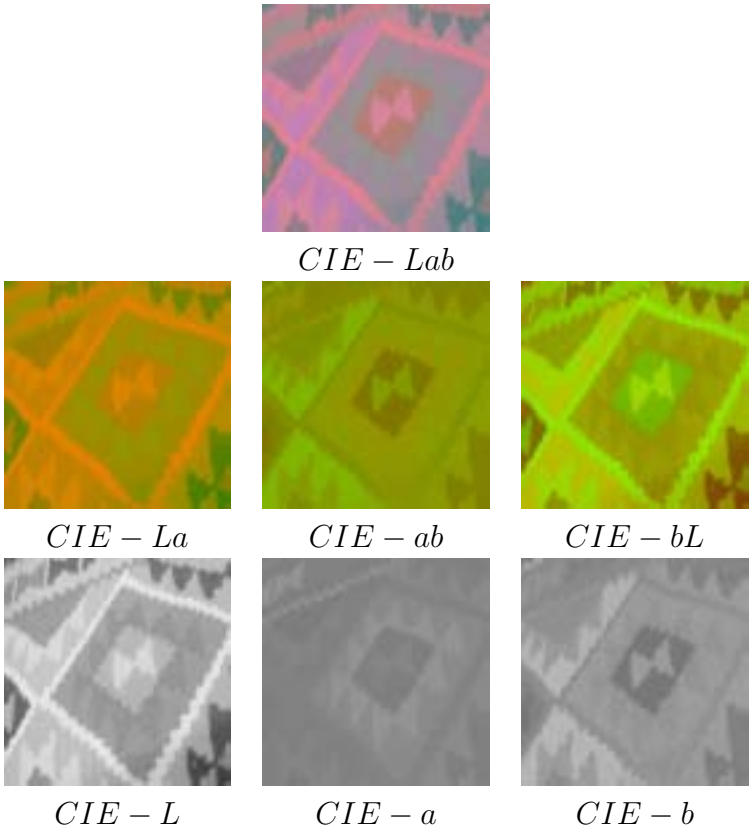


Figure 2.12: Representation of the sample image in $CIE - Lab$ color space.

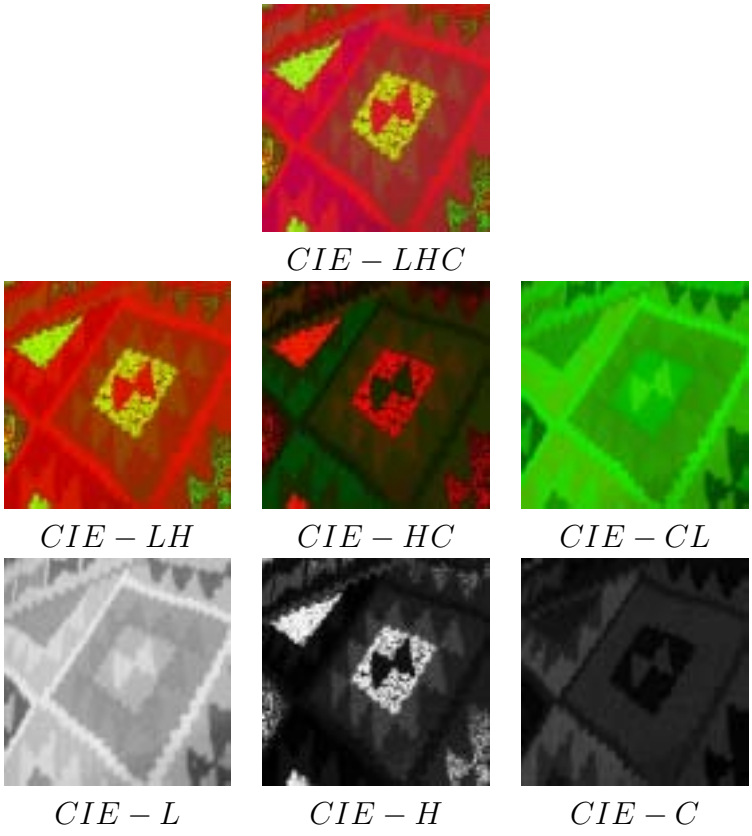


Figure 2.13: Representation of the sample image in $CIE - LHC$ color space.

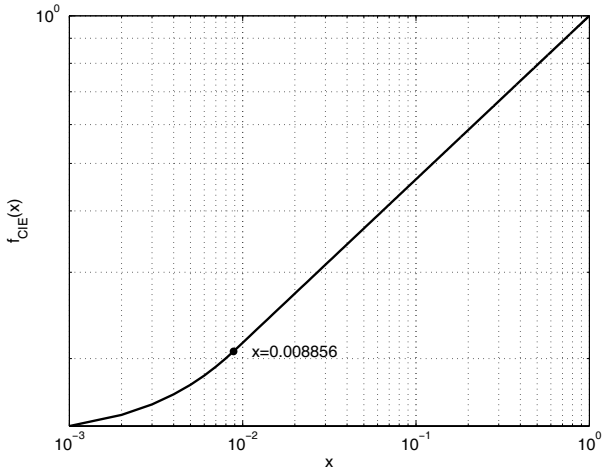


Figure 2.14: Values of the function f_{CIE} defined in (2.1.13).

Figure 2.14 shows the values of $f_{CIE}(x)$ for $x \in [0, 1]$

There are some less–famous color spaces, which are devised for special purposes. For example, the IR_gB_y color space and its variant $IR_gB_y^+$ are used in pornography filtering [140].

$$IR_gB_y : \begin{cases} I = L(G) \\ R_g = L(R) - L(G) \\ B_y = L(B) - \frac{L(R)+L(G)}{2} \\ L(x) = 105 \log_{10}(x + 1) \end{cases}, \tag{2.1.14}$$

$$IR_gB_y^+ : \begin{cases} I = L(G) \\ R_g = L(R) - L(G) \\ B_y = L(B) - \frac{L(R)+L(G)}{2} \\ L(x) = 105 \log_{10}(x + 1 + n) \end{cases}, \tag{2.1.15}$$

where, n is a uniform random variable in $[0, 1]$. Figures 2.15 and 2.16 show the

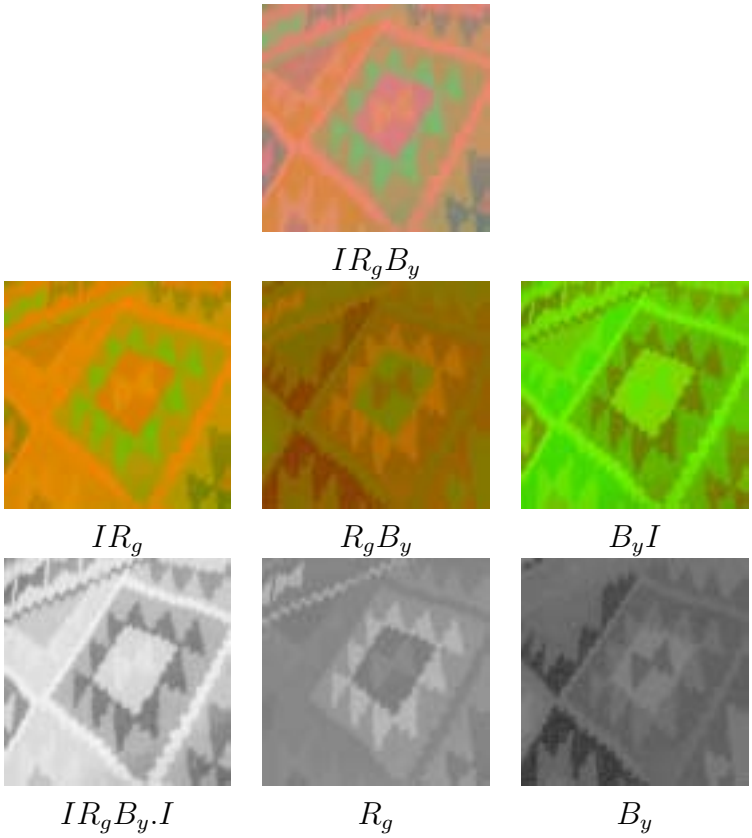


Figure 2.15: Representation of the sample image in IR_gB_y color space.

representation of the sample image in IR_gB_y and $IR_gB_y^+$ color spaces, respectively.

Also, $Nrgb$ is the normalized version of the RGB color space [321].

$$Nrgb : \begin{cases} r = \frac{R}{R+G+B} \\ g = \frac{G}{R+G+B} \\ b = \frac{B}{R+G+B} \end{cases} . \tag{2.1.16}$$

Note that $Nrgb$ is essentially a 2–D color space. Figure 2.17 shows the sample image represented in this color space.

Some researchers tend to convert cartesian coordinates to polar ones [195] and

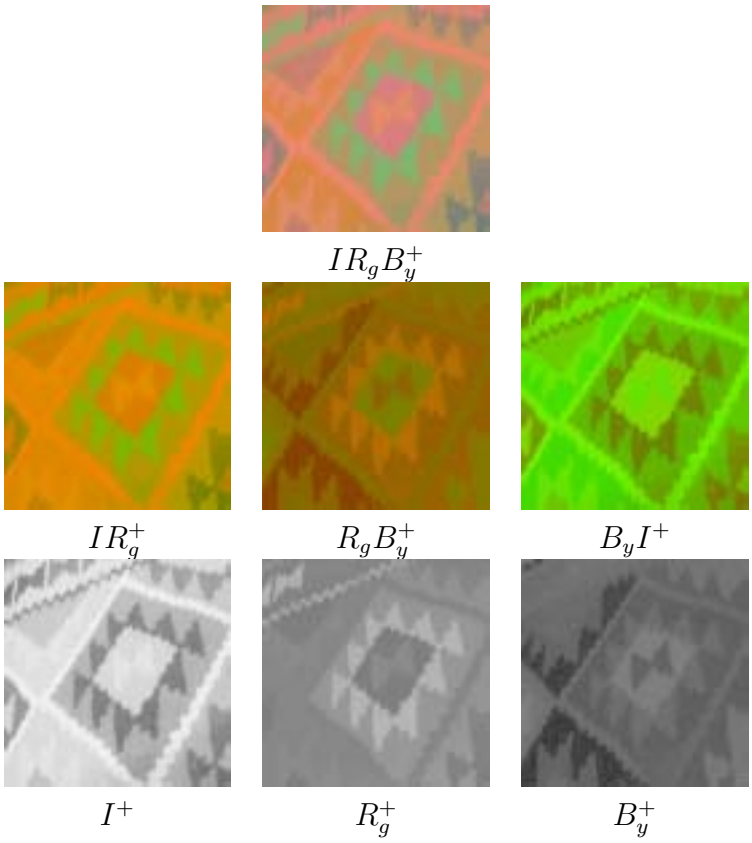


Figure 2.16: Representation of the sample image in $IR_gB_y^+$ color space.

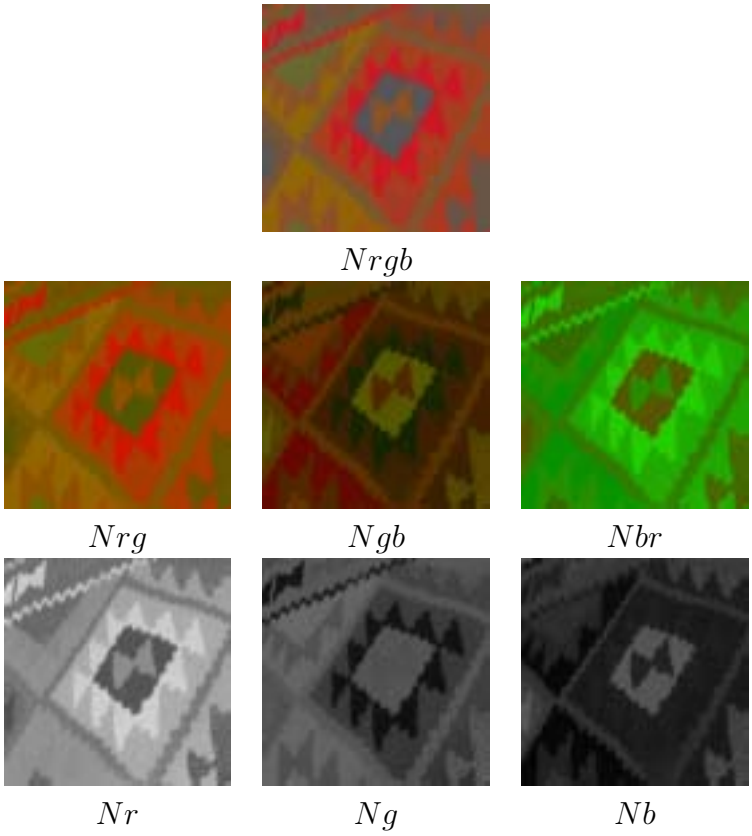


Figure 2.17: Representation of the sample image in $Nrgb$ color space.

vice versa [72]. As such, $YC_bC_r^\circ$, YIQ°, and YUV° are defined as the polar version of YC_bC_r, YIQ, and YUV, respectively. Also, HSI^c and HSV^c are the cartesian versions of HSI and HSV, respectively.

$$YC_bC_r^\circ : \begin{cases} Y = 0.30R + 0.59G + 0.11B \\ I^\circ = \sqrt{C_b^2 + C_r^2} \\ Q^\circ = \tan^{-1} \frac{C_b}{C_r} \end{cases}, \tag{2.1.17}$$

$$YIQ^\circ : \begin{cases} Y^\circ = 0.30R + 0.59G + 0.11B \\ C_b^\circ = \sqrt{I^2 + Q^2} \\ C_r^\circ = \tan^{-1} \frac{I}{Q} \end{cases}, \tag{2.1.18}$$

$$YUV^\circ : \begin{cases} Y = 0.30R + 0.59G + 0.11B \\ U^\circ = \sqrt{U^2 + V^2} \\ V^\circ = \tan^{-1} \frac{U}{V} \end{cases}, \tag{2.1.19}$$

$$HSI^c : \begin{cases} H^c = S \sin H \\ S^c = S \cos H \\ I^c = \frac{1}{3}(R + G + B) \end{cases}, \tag{2.1.20}$$

$$HSV^c : \begin{cases} H^c = S \sin H \\ S^c = S \cos H \\ V^c = max(R, G, B) \end{cases}. \tag{2.1.21}$$

Figures 2.18, 2.19, 2.20, 2.21, and 2.22 visualize the representation of the sample image in $YC_bC_r^\circ$, YIQ°, YUV°, HSI^c and HSV^c, respectively.

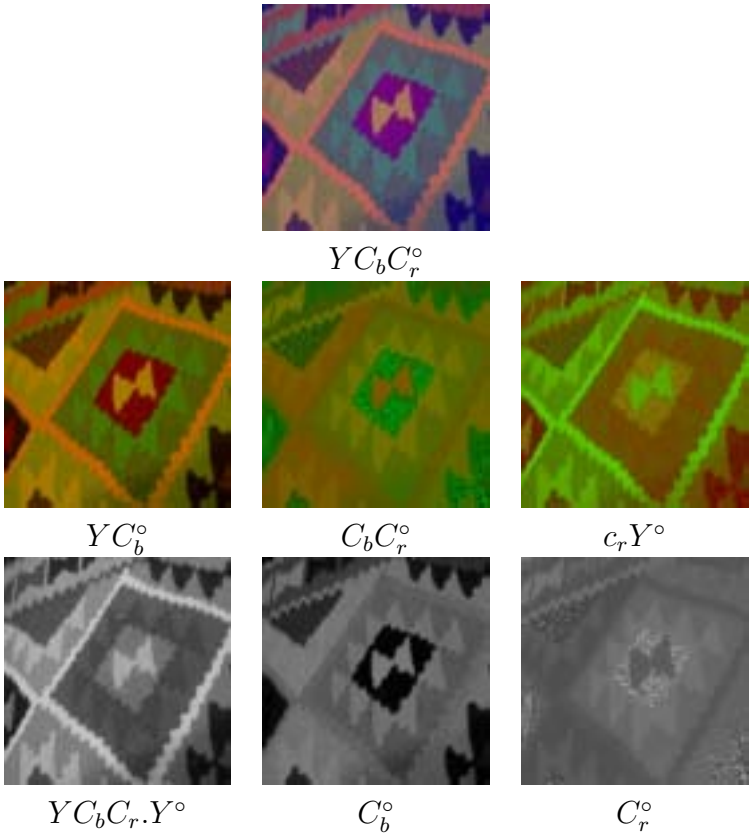


Figure 2.18: Representation of the sample image in $YC_bC_r^\circ$ color space.

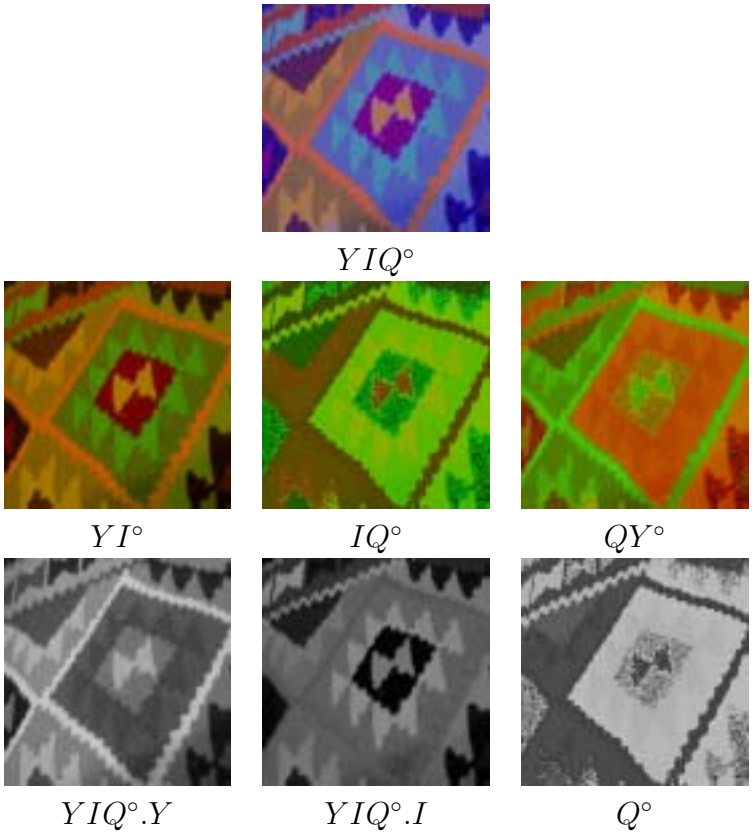


Figure 2.19: Representation of the sample image in YIQ° color space.

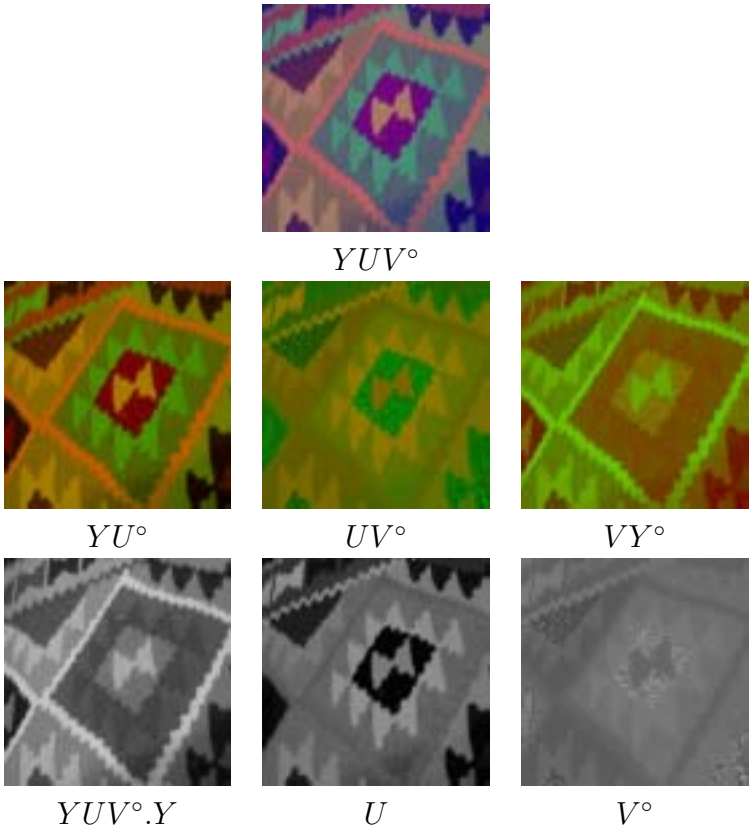


Figure 2.20: Representation of the sample image in YUV° color space.

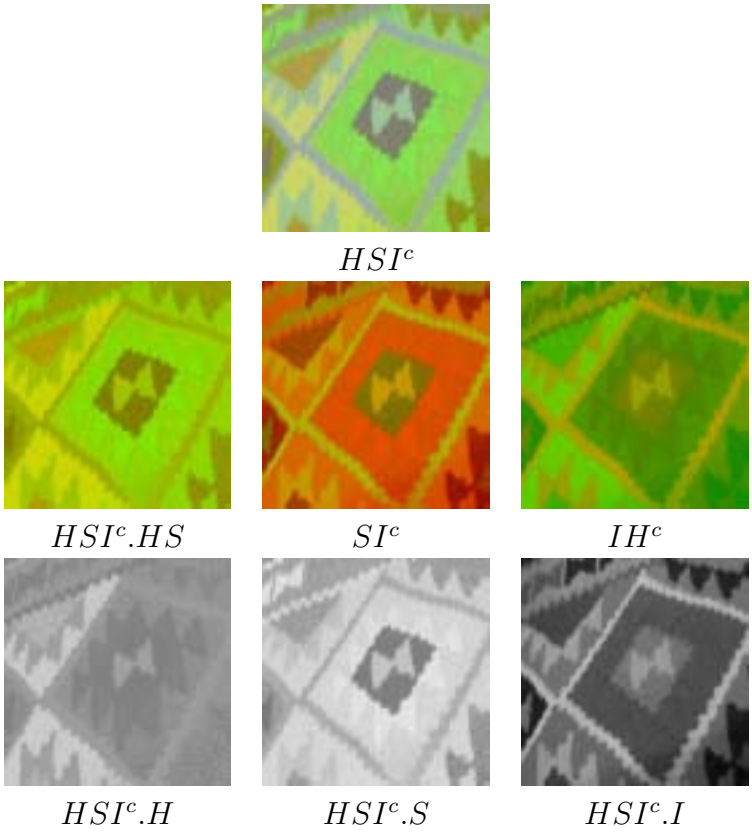


Figure 2.21: Representation of the sample image in HSI^c color space.

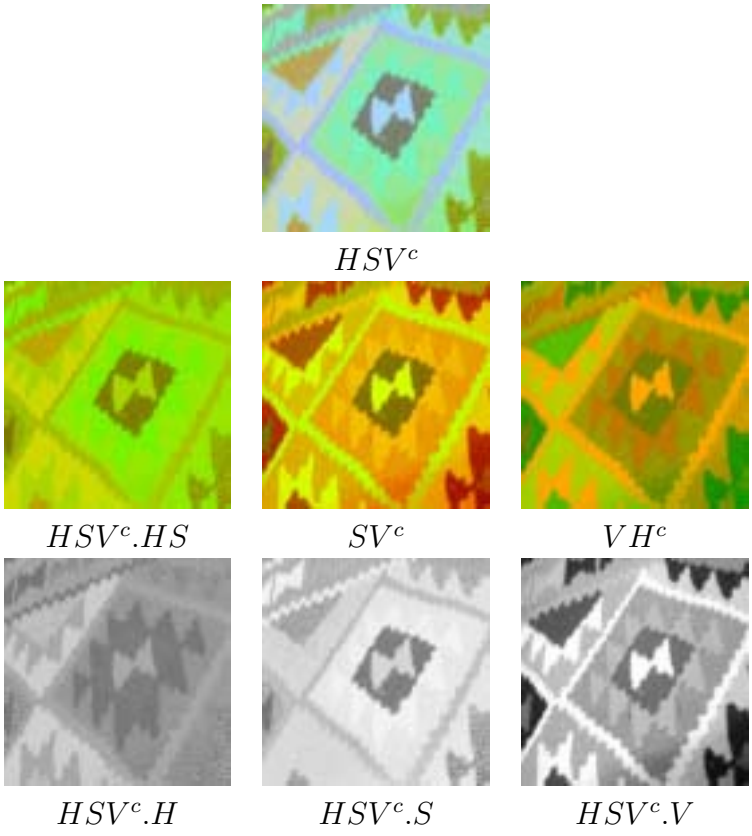


Figure 2.22: Representation of the sample image in HSV^c color space.

The ratio RGB is defined as,

$$RGB^r : \begin{cases} R^r = \frac{R}{G} \\ G^r = \frac{G}{B} \\ B^r = \frac{B}{R} \end{cases}, \tag{2.1.22}$$

which is a 2–D color space. Finally, Tint–Saturation–Lightness (TSL) color space is defined as [352],

$$TSL : \begin{cases} T = \begin{cases} \frac{1}{2\pi}\tan^{-1}\frac{r'}{g'} + \frac{1}{4}, & g' < 0 \\ \frac{1}{2\pi}\tan^{-1}\frac{r'}{g'} + \frac{3}{4}, & g' > 0 \\ 0, & g' = 0 \end{cases} \\ S = \sqrt{\frac{9}{5}(r'^2 + g'^2)} \\ L = 0.30R + 0.59G + 0.11B \\ r' = r - \frac{1}{3} \\ g' = g - \frac{1}{3} \end{cases}, \tag{2.1.23}$$

where r and g come from (2.1.16). Figures 2.23 and 2.24 show the representation of the sample image in RGB^r and TSL color spaces.

The interested reader is referred to the review of color spaces in [287]. Also, see [64] for a historical review and psychological discussions and [133] for constructive ideas. Reviewing the bibliography of this thesis reveals that there is no specific application of color image processing in which researchers agree about the choice of the best color space. We refer the reader to [139] for further information about color spaces.

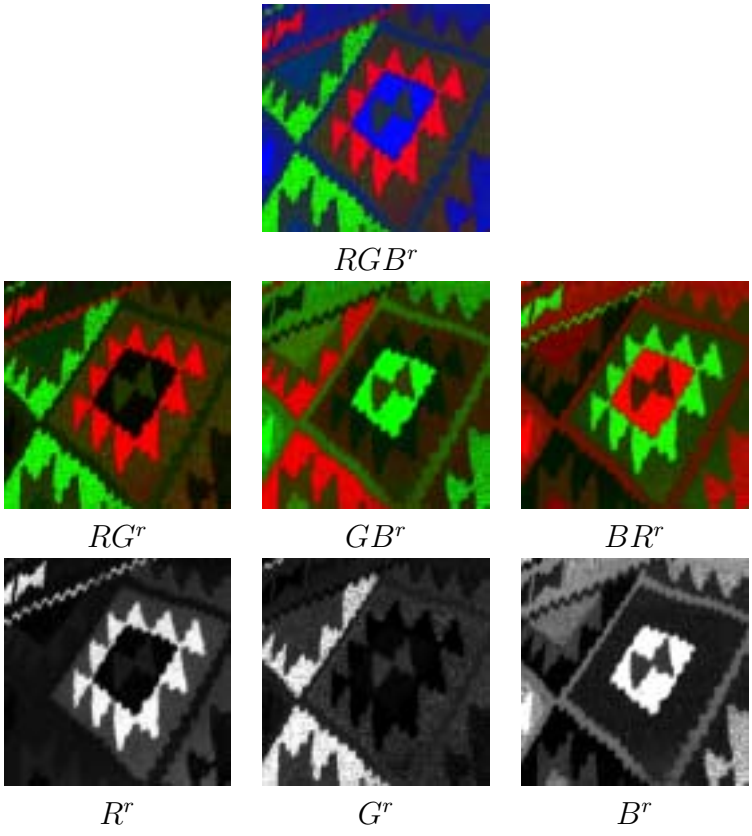


Figure 2.23: Representation of the sample image in RGB^r color space.

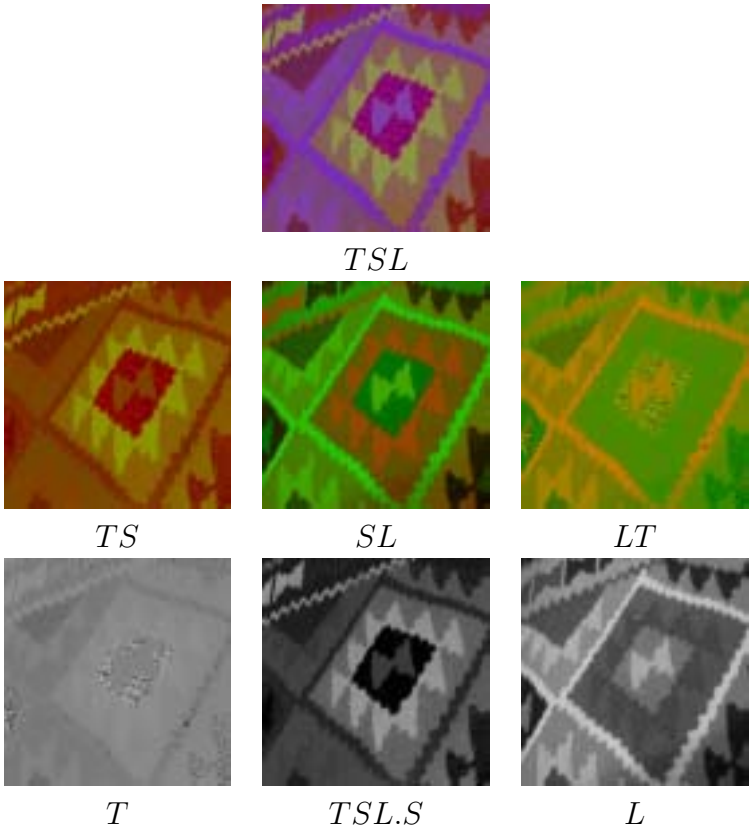


Figure 2.24: Representation of the sample image in TSL color space.

2.1.2 Implementation in CIP

Almost all color spaces are (mostly, one–to–one) functions $F : (N_{255} \cup \{0\})^3 \rightarrow R^3$. For convenient, we call the set $(N_{255} \cup \{0\})^3$ as Θ. In order to make the representation of different color spaces comparable, we perform a linear normalization scheme. In this way, for each color space $F = [F_1, F_2, F_3]^T$, $(F_i : \Theta \rightarrow R)$, the normalized color space $\tilde{F}$ is defined as,

$$\tilde{F}(\vec{c}) = \begin{pmatrix} [\alpha_1 F_1(\vec{c}) + \beta_1] \\ [\alpha_2 F_2(\vec{c}) + \beta_2] \\ [\alpha_3 F_3(\vec{c}) + \beta_3] \end{pmatrix}, \tag{2.1.24}$$

where, α_i and β_i are constant real values and $[x]$ denotes the integer value of x. The constants α_i and β_i are selected in the way that $\tilde{F}$ turns into a $\Theta \rightarrow \Theta$ function, which is applicable for digital 8–bit processing. Table 2.1 lists the values of α_i and β_i for the color spaces stated above. For a color component F we have,

$$\alpha = \frac{255}{\max_{\vec{x} \in \Theta}\{F(\vec{x})\} - \min_{\vec{x} \in \Theta}\{F(\vec{x})\}}, \tag{2.1.25}$$

$$\beta = 255 \frac{-\min_{\vec{x} \in \Theta}\{F(\vec{x})\}}{\max_{\vec{x} \in \Theta}\{F(\vec{x})\} - \min_{\vec{x} \in \Theta}\{F(\vec{x})\}}. \tag{2.1.26}$$

In some cases that computing the global maximum and minimum values of a single color component was not practically possible, a fair approximation is used.

The naming of many color spaces makes a difficulty in calling them. For example, the color space Rg may be considered as the 2–D subspace of RGB and also as the 1–D subspace of IR_gB_y (this problem is serious when using the names of color spaces for naming files). Hence, a new naming convention is proposed in this thesis. As such,

Table 2.1: Normalization factors for the color spaces under investigation.

Color Space	α_1	α_2	α_3	β_1	β_2	β_3
RGB	1	1	1	0	0	0
$HMMD$	$255/\pi$	1	1	0	0	0
$I_1I_2I_3$	1	1	1	255/2	255/2	0
HSI	$255/\pi$	255	1	0	0	0
$CIE-XYZ$	0.84	1	1	0	0	0
HSV	$255/(2\pi)$	255	1	0	0	0
YC_bC_r	1	0.56	0.71	0	255/2	255/2
$CIE-Lab$	265.8	147.9	147.9	-40.8	255/2	255/2
YIQ	1	0.83	0.96	0	255/2	255/2
$CIE-LHC$	2.55	$-255/\pi$	$255/\sqrt{2}$	-40.8	255/2	0
YUV	1	1.14	0.81	0	255/2	255/2
$CIE-Luv$	2.55	0.58	0.78	-40.8	94	140
IR_gB_y	1	0.5	0.5	0	255/2	255/2
$IR_gB_y^+$	1	0.5	0.5	0	255/2	255/2
$Nrgb$	255	255	255	0	0	0
YUV°	1	1.32	$255/(2\pi)$	0	0	255/2
YIQ°	1	1.26	$255/(2\pi)$	0	0	255/2
$YCbCr^\circ$	1	0.88	$255/(2\pi)$	0	0	255/2
HSI^c	1	255/2	255/2	0	255/2	255/2
HSV^c	1	255/2	255/2	0	255/2	255/2
RGB^r	–[a]	–	–	–	–	–
TSL	$2 \times 255/3$	$255\sqrt{5}/\sqrt{8}$	1	255/6	0	0

[a]Range of this color space spans $[0, \infty]$. As no global normalization works on RGB^r, its components are locally (in each image) normalized.

Table 2.2: Naming convention for the color spaces and their 2–D representations.

Color Space	x_1x_2	x_2x_3	x_3x_1
$HMMD$	$HMin$	$MinMax$	$MaxH$
$I_1I_2I_3$	I_1I_2	I_2I_3	I_3I_1
HSI	$HSI.HS$	SI	IH
$CIE-XYZ$	$CIE-XY$	$CIE-YZ$	$CIE-ZX$
HSV	$HSV.HS$	SV	VH
YC_bC_r	YC_b	C_bC_r	C_rY
$CIE-Lab$	$CIE-La$	$CIE-ab$	$CIE-bL$
YIQ	$YIQ.YI$	IQ	QY
$CIE-LHC$	$CIE-LH$	$CIE-HC$	$CIE-CL$
YUV	YU	$YUV.UV$	VY
$CIE-Luv$	$CIE-Lu$	$CIE-uv$	$CIE-vL$
IR_gB_y	IR_g	R_gB_y	B_yI
$IR_gB_y^+$	IR_g^+	$R_gB_y^+$	B_yI^+
$Nrgb$	Nrg	Ngb	Nbr
YUV°	YU°	UV°	VY°
YIQ°	YI°	IQ°	QY°
$YC_bC_r^\circ$	YC_b°	$C_bC_r^\circ$	C_rY°
HSI^c	$HSI^c.HS$	SI^c	IH^c
HSV^c	$HSV^c.HS$	SV^c	VH^c
RGB^r	RG^r	GB^r	BR^r
TSL	TS	SL	LT

when there is the possibility of misunderstanding, the name of the mother space is added to its children. Tables 2.2 and 2.3 lists the naming convention for different color spaces used in this thesis.

The function f_{CIE} defined in (2.1.13) is computed by the function *cief* for the arbitrary–sized arrays of type *double* (see Figure 2.25).

To get aware of the list of color spaces supported in **CIP**, one may execute the script *colorspaces*. Typing "*help colorspaces*" yields the same result. As such, a

Table 2.3: Naming convention for the color spaces and their 1–D representations.

Color Space	x_1	x_2	x_3
$HMMD$	$HMMD.H$	Min	Max
$I_1I_2I_3$	I_1	I_2	I_3
HSI	$HSI.H$	$HSI.S$	$HSI.I$
$CIE-XYZ$	$CIE-X$	$CIE-Y$	$CIE-Z$
HSV	$HSV.H$	$HSV.S$	$HSV.V$
YC_bC_r	$YC_bC_r.Y$	C_b	C_r
$CIE-Lab$	$CIE-L$	$CIE-a$	$CIE-b$
YIQ	$YIQ.Y$	$YIQ.I$	Q
$CIE-LHC$	$CIE-L$	$CIE-H$	$CIE-C$
YUV	$YUV.Y$	$YUV.U$	$YUV.V$
$CIE-Luv$	$CIE-L$	$CIE-u$	$CIE-v$
IR_gB_y	$IR_gB_y.I$	R_g	B_y
$IR_gB_y^+$	I^+	R_g^+	B_y^+
$Nrgb$	Nr	Ng	Nb
YUV°	$YUV^\circ.Y$	U°	V°
YIQ°	$YIQ^\circ.Y$	$YIQ^\circ.I$	Q°
$YC_bC_r^\circ$	$YC_bC_r^\circ.Y$	C_b°	C_r°
HSI^c	$HSI^c.H$	$HSI^c.S$	$HSI^c.I$
HSV^c	$HSV^c.H$	$HSV^c.S$	$HSV^c.V$
RGB^r	R^r	G^r	B^r
TSL	T	$TSL.S$	L

cief

```
function y=CIEf(x)
%Function CIE function, computes the CIE transformation used in the CIE
%defined color spaces of CIE-Lab, CIE-luv, and  CIE-LHC.
%Usage:      y=CIEf(x)
%Inputs:     x:the input.
%Outputs:    y:The result.
%Refer to
%   A. Abadpour, S. Kasaei, A new parametric linear adaptive color space and its
%   pca–based implementation, in: The 9th Annual CSI Computer Conference,
%   CSICC, Tehran, Iran, Feb. 2004, pp. 125–132.
%Arash Abadpour, Revision 1.01, Winter 2005 (Jan. 15, 2005).
```

Figure 2.25: Implementation of the *cief* function.

list of the color spaces available in **CIP** will be shown on the display. Figure 2.26 shows this function. Also, function *csmax* gives the maximum index of the available color spaces. Using the function *cname* one can retrieve the name of a color space associated with an index (see Figure 2.28). A typical set of arguments to the *csname* function includes an index (in the range specified by *csmax*), a boolean (specifying wether the string should be LaTeX 2_ε–compliant or a normal text), and a subspace specifier. The subspace specifier which is a member of $\{123, 12, 23, 31, 1, 2, 3\}$, denotes the subspace. Figure 2.29 shows some typical results of the *csname* function.

For any color space supported in **CIP**, there exists a function named *rgb2*$\star$. Figure 2.30 shows the template of these functions. Any *rgb2*$\star$ function inputs an image and a boolean. The boolean argument specifies wether the function should normalize

colorspaces

```
%File colorspaces.m holds the names of the available color spaces.
%The available color spaces are :
%  0 -RGB
%  1 -HMMD
%  2 -I1I2I3
%  3 -HSI
%  4 -CIE-XYZ
%  5 -HSV
%  6 -YCbCr
%  7 -CIE-Lab
%  8 -YIQ
%  9 -CIE-LHC
%  10-YUV
%  11-CIE-Luv
%  12-IRgBy
%  13-IRgBy+
%  14-Nrgb
%  15-YUVo
%  16-YIQo
%  17-YCbCro
%  18-HSIc
%  19-HSVc
%  20-RGBr
%  21-TSL
%Arash Abadpour Revision 1.02 Spring 2005 (April 13, 2005).
```

Figure 2.26: Implementation of the *colorspaces* function.

csmax

```
function i=csmax
%Function maximum color space return the index to the last color space
%currently available in the CIP toolbox.
%Usage:      i=csmax;
%Inputs:
%Outputs:    i:Index (type help colorspaces to see the names of the available
%             color spaces).
%Arash Abadpour Revision 1.0 Spring 2005 (March 25, 2005).
```

Figure 2.27: Implementation of the *csmax* function.

csname

```
function s=csname(index,TX,i)
%Function color space name returns thename of the colorspace corresponding
%to index.
%Usage:      s=csname(index,TX,i)
%Inputs:     index: type help ColorSpaces
%            TX: LATEX Format?
%            i:Name what? (123,12,23,31,1,2,3)
%Outputs:    s:Color space name.
%Refer to
%   A. Abadpour, S. Kasaei, A new parametric linear adaptive color space and its
%   pca–based implementation, in: The 9th Annual CSI Computer Conference,
%   CSICC, Tehran, Iran, Feb. 2004, pp. 125–132.
%Arash Abadpour Revision 2.131 spring 2005 (April 11, 2005).
```

Figure 2.28: Implementation of the *csname* function.

```
csname(2)="I_1I_2I_3"
csname(3,0)="HSI"
csname(8,0,12)="YIQ.YI"
csname(12,1,3)="B_y"
```

Figure 2.29: Sample results of the *csname* function.

the results (see Table 2.1 for details). Table 2.4 shows the list of color space conversion functions present in **CIP**.

CIP contains two functions *rgb2x* and *x2rgb* which perform the conversion when given the index.The functions inputs the index to the color space and utilize it (see Figures 2.31 and 2.32). As shown in Figure 2.31, the function *rgb2x* inputs an image plus an index (to a color space). Also, *rgb2x* gives the normalization indicator and a specifier to produce subspaces.

From the above discussion it derives that **CIP** professionally supports color space evaluation. Using the functions *csmax*, *csname*, and *rgb2x*, one is able to perform an identical task using different color spaces. This scheme is frequently used in this thesis.

An attempt to add a new color space to **CIP** is straight forward. First, a function named *rgb2*$\star$ should be developed. Here, $\star$ denotes the name of the new color space. The function should comply with the template shown in Figure 2.30. Then, after associating the first available index to the new color space, its link should be added to *rgb2x* and *csname*. Also, the maximum index in *csmax* have to be updated. In

templatex2y

```
function J=x2y(I,N)
%Function x2y, converts the coordinates from the X color space to
%the Y color space.
%Usage:      J=x2y(I,N);
%Inputs:     I:The original image.
%            N:Normalize?
%Outputs:    J:The result.
%Arash Abadpour Revision 2.0 Spring 2005 (April 13, 2005).

if nargin<2
    N=1;
end

[R,G,B]=splitrgb(I);

%The formulation
r=R./(G+eps);
g=G./(B+eps);
b=B./(R+eps);

%The normalization
if N==1
    r=r*255;
    g=g*255;
    b=b*255;
end

J=mergergb(r,g,b);
```

Figure 2.30: Implementation of the *templatex2y* function.

Table 2.4: Color space conversion functions present in **CIP**.

From	**To**	**Function Name**
RGB	$HMMD$	*rgb2hmmd*
RGB	$I_1I_2I_3$	*rgb2ohta*
RGB	HSI	*rgb2hsi*
RGB	$CIE-XYZ$	*rgb2xyz*
RGB	HSV	*rgb2hsv*
RGB	YC_bC_r	*rgb2ycbcr*
RGB	$CIE-Lab$	*rgb2lab*
RGB	YIQ	*rgb2yiq*
RGB	$CIE-LHC$	*rgb2lhc*
RGB	YUV	*rgb2yuv*
RGB	$CIE-Luv$	*rgb2luv*
RGB	IR_gB_y	*rgb2IRgBy*
RGB	$IR_gB_y^+$	*rgb2IRgByp*
RGB	$Nrgb$	*rgb2Nrgb*
RGB	YUV°	*rgb2yuvo*
RGB	YIQ°	*rgb2yiqo*
RGB	$YC_bC_r^\circ$	*rgb2ycbcro*
RGB	HSI^c	*rgb2hsic*
RGB	HSV^c	*rgb2hsvc*
RGB	RGB^r	*rgb2rgbr*
RGB	TSL	*rgb2tsl*
$I_1I_2I_3$	RGB	*ohta2rgb*
$CIE-XYZ$	RGB	*xyz2rgb*
YC_bC_r	RGB	*ycbcr2rgb*
YIQ	RGB	*yiq2rgb*
YUV	RGB	*yuv2rgb*

rgb2x

```
function J=rgb2X(I,index,N,i)
%Function rgb2X, converts the coordinates from the rgb color space to
%the selected color space.
%Usage:      J=rgb2X(I,index,N,i)
%Inputs:     I:The original image.
%           index: type help ColorSpaces
%           N:Normalize?
%           i:Produce what? (123,12,23,31,1,2,3)
%Outputs:    J:The result.
%Refer to
%   A. Abadpour, S. Kasaei, A new parametric linear adaptive color space and its
%   pca–based implementation, in: The 9th Annual CSI Computer Conference,
%   CSICC, Tehran, Iran, Feb. 2004, pp. 125–132.
%Arash Abadpour Revision 1.5 Spring 2005 (April 11, 2005).
```

Figure 2.31: Implementation of the *rgb2x* function.

x2rgb

```
function J=X2rgb(I,index)
%Function X2rgb, converts the coordinates from the selected color space to
%the RGB color space.
%Usage:      J=X2rgb(I,index,N);
%Inputs:     I:The original image.
%           index: type help ColorSpaces (many color spaces are not supported).
%Outputs:    J:The result.
%Refer to
%   A. Abadpour, S. Kasaei, A new parametric linear adaptive color space and its
%   pca–based implementation, in: The 9th Annual CSI Computer Conference,
%   CSICC, Tehran, Iran, Feb. 2004, pp. 125–132.
%Arash Abadpour Revision 1.1 Spring 2005 (April 13, 2005).
```

Figure 2.32: Implementation of the *X2rgb* function.

this stage, one may decide to develop the inverse function *⋆2rgb*, too. The link to this function should inserted into *x2rgb*. Now, the new color space is correctly embedded into **CIP**.

2.2 Color Image Processing

In this section we briefly review the available color image processing literature. The review includes those fields covered in this thesis.

2.2.1 Spotting

The first step towards recognizing an object in a captured image is to distinguish it from the background. An object–in–image segmentation process is meant to detect the area containing the object; and to extract the edges between the object and the background. When using such method, the result may include some parts of other objects. This is sometimes inevitable but for performance reasons, an accurate segmentation task is preferred which completely removes the background. It must be denoted that segmentation processes, that work on color information of each pixel independent of the neighborhood information are more preferred for their byte–stream tendency and algorithm speed.

Spotting colors concerns with the accuracy with which objects of a specific single color can be identified in a complex image [248]. Although many color–based object

recognition methods exist [343, 127, 401, 146], generally they work on a multicolored object. For example, *Yullie et al.* [401] introduced an algorithm for detecting street signs. The method uses the relative appearance of the two colors in the signs. Algorithms proposed by *Ennesser et al.* [127] and *Funt et al.* [146] use color–edge histograms for recognizing a multi–colored object.

2.2.2 Distance Measures

The Euclidean distance is generally accepted as a proper likelihood measure in color fields [280, 84, 254, 246, 97, 247, 112, 237]. For example in [247], the authors use the Euclidean distance in the $CIE - Lab$ color space. The Euclidean distance only depends on the central point of the cluster (and not on its shape). Although, its theoretically simple, it is not a perfect solution. To overcome the shortcomings of the Euclidean distance–based likelihood measures, in [395] the authors used a weighted Euclidean distance, which when tuned well converges to the well–known Mahalanobis distance. In that approach, not only the central point of the cluster is considered, but also the spread of the data is incorporated as a set of normalizing factors and a rotation matrix.

2.2.3 Fuzzification Function

As the likelihood measures tend to compute the distance between a point and a cluster, they rank better members with smaller values. This is in contrast with

the fuzzyfication approach, which marks better members with larger values (in the range of $[0, 1]$). Thus, a mapping function is needed to produce the desired result. The general definition of such a mapping is $y = f(\frac{x}{\sigma})$, where f is an even function, mapping $[0, \infty]$ to $[0, 1]$ in an upside–down scheme. Considering a cluster r, on which the distance function is defined as $\tau_r(\vec{c})$, leads to the likelihood measure of $f(\frac{\tau_r(\vec{c})}{\sigma})$. For all $\vec{c} \in r$, we expect the likelihood measure to be almost unity. Thus, σ must be statistically computed in terms of the members of the set $\{\tau_r(\vec{c})|\vec{c} \in r\}$. Also, f must have a flat top, because we expect f to map the interval $[-1, 1]$ to the vicinity of unity. This is in contrast with the generally–used Gaussian functions [97].

2.2.4 Clustering

As a non–hierarchical clustering method, fuzzy clustering, is proved to be efficient in clustering a set of given vectors into a few homogenous groups [334]. In conventional clustering methods, samples are either accepted or rejected to be assigned to each class. In contrast, the fuzzy clustering methods incorporate the membership concept into the clustering results. The fuzzy clustering is becoming more popular because it can produce the crisp results when needed. For example in [253], *Massone et. al.* use the relabelling algorithm [361], to produce crisp clusters after performing a fuzzy clustering method on remote sensing image data. Also, fuzzy clustering is less prone to falling into local optima than the crisp clustering algorithms [187, 61, 188].

The idea of fuzzy clustering came from the *Hard C-Means* (HCM) method proposed by *Ruspini* (1969) [306, 124]. He introduced a notion of fuzzy partition, to describe the cluster structure of a data set and suggested an algorithm to compute the optimum fuzzy partitioning. *Dunn* (1973) [125] generalized the minimum–variance clustering procedure into a Fuzzy *Iterative Self Organizing Data Analysis Technique* (ISODATA) clustering technique. *Bezdek* (1981) [60] generalized *Dunn*'s approach to obtain an infinite family of algorithms which is called the *Fuzzy C-Means* (FCM). The main contribution of *Bezdek* is the introduction of the concept of fuzziness (m) [168]. An extension to the FCM is the *Gustafson–Kessel* (GK) method [161], which uses the covariance matrix of data to capture ellipsoidal properties of the clusters. The novel idea of the GK method is using the Mahalanobis distance. After that, the *Gath–Geva* method (GG) used the same distance [148]. Other contributions in this field, include the *fuzzy c–varieties* (FCV) and *fuzzy c–elliptotypes* (FCE) [60, 236], fuzzy c-spherical shells [226, 115], clustering algorithms based on volume criteria [225], and fuzzy and possibilistic shell clustering [221, 222]. In 1993, *Krishnapuram* and *Keller* proposed the *probabilistic fuzzy C–means* (PCM) clustering method [223, 224]. Although, PCM adds more noise robustness to the FCM, it uses the same definition of Euclidean distance between the points and the clusters.

There are three major difficulties in fuzzy clustering [186]:

1. Determining the number of clusters.

2. Initialization.

3. Handling cluster characteristics (like its shape).

For example, in [212] the authors use an Euclidean–based initialization scheme for color image segmentation. *Bezdek* reported the three most popular initialization techniques as follows [61]:

1. Random initialization [202, 57, 100, 320].

2. Using the first distinct data points [353, 235].

3. Using uniformly distributed data points [186].

The performance of the FCM is reported in many studies. In [359, 276, 187, 61] the researchers used the FCM for multivariate image segmentation and in [100] *Cheng et. al.* present significantly faster FCM using a multistage random sampling model. In [186] *Iyer et. al.* applied the FCM to detect the blurred mammogram in low contrast images. Also, FCM is used for object detection [320] and color quantization [295].

2.2.5 Tree Decomposition

Quad-tree decomposition is the well–known method for splitting an image into homogenous sub-blocks, resulting in a very coarse but fast segmentation [311]. Defining

the whole image as a single block, the method is performed according to a problem-specific homogeneity criterion. Some works on the generalization of quad-tree decomposition is performed, regarding the dimension [128] and the shape [394] of the blocks. Using rectangles as the sub–blocks is known to have many benefits. Firstly, performing block–wise operations in rectangular regions is computationally efficient. Also, more complicated shapes of blocks (such as triangles) perform division operator on spatial variables which leads to more round-off and misalignment errors. In the conventional quad–tree decomposition method, when a non–homogenous block is encountered, it is split into four sub–blocks. The one–split–to–four rule is optimized in this thesis by using a novel one–split–to–two rule, resulting in a method called bi–tree decomposition.

2.2.6 Query Region Extraction

Query region extraction is the process of finding the region similar to a given query region in an image [89]. The similarity is defined in terms of color, texture, and the forth. Query region extraction is used in many computer vision applications like image and video segmentation [362], face detection [332, 47, 143, 208, 88], region–based coding [84], and so on. Also, it is used in computer graphics applications such as alpha–estimation [135, 137, 261, 308], colorizing gray–scale images [375, 387], and recoloring color images [297, 92].

There exists many crisp approaches to color region extraction with more or less

satisfactory results (*e.g.*, see [343, 102, 180, 118, 52]). But it is proved that the fuzzy approach is more capable of facing the tolerances and characteristics of image segmentation [61]. There are a few available works on fuzzy region extraction [313, 366, 142, 89], which they all use histogram dissimilarity measures in a very expensive window sliding framework.

2.2.7 Segmentation

Color image segmentation is an spatial–spectral problem: The desired result is a clustering of the color histogram that leads to spatially contiguous, minimally–overlapping regions [246, 167, 144, 301, 276, 288]. Due to this nature of the problem, the solutions to this problem can be categorized into three classes of spectral [415, 254, 74, 97, 246], spatial [246] and hybrid [380]. In the spectral class, one of the general–purpose unsupervised clustering methods like the ISODATA [68], the *Information Bottleneck*(IB) [354], PCA [183], Thresholding, or fuzzy clustering [254] is used for clustering the color histogram. In another stage, this clustering is applied on the image and some alterations is performed on clusters to best fit the problem. In the spatial algorithms, the process begins with working on spatial information using region growth [301] or decomposition methods [311]. The hybrid algorithms that perform on spectral–spatial or spatial–spectral order, use both information simultaneously. For example in [380] the author uses a double mask and produces a 9×1 vector corresponding to each pixel. This vector contains both spectral and spatial information.

It is well–known that the methods that insist on spectral information are faster while produce more ragged regions [246]. The spatial methods are in return producing better regions while needing more time [246]. The main problem with spectral methods is their lack of knowledge about neighborhoods, the problem known as surface coherence [190] (for a comprehensive review see [246]).

2.2.8 Principal Colors

In 1988, *Klinker*, *Shafer*, and *Kanade* presented a novel approach to measurement of highlights in color images [219]. There, they developed a proper model for the reflected light from an arbitrary point of a dielectric object. In 1990, they applied their approach to color image understanding [220]. Though, more than a decade passed since the idea was successfully incorporated into a practical algorithm. In 2003, without being seriously involved in the theoretical aspects, *Cheng* and *Hsia* used *principal component analysis* (PCA) for color image processing [99]. Then in 2004, *Nikolaev* and *Nikolayev* started the work again from the theory and proved that PCA is a proper tool for color image processing [268]. The next necessary step was introduced in the early 1991, when *Turk* and *Pentland* proposed their eigenface method [363]. There, they developed a novel idea which connected the eigenproblems in the color domain and the spatial domain. Although, there is this rich theoretical background for the linear local models of color constancy, it is quite common to see research procedures which are based on the old color space paradigm, even in 2005.

In the PCA framework, each color swatch is represented by a line (passing through the expectation vector, parallel to the principal direction). Although it is proved that the lines representing similar material lie on a unique plane [219], the theory neglects the (maybe virtual) intersection of these lines. In this thesis we show that the lines of similar material intersect and show that this intersection is a proper tool for color image segmentation. We call this intersection point as the principal color of the undergoing material. Everywhere in this thesis, an intersection of a set of points is a point close to which all line traverse.

Even a careless review of the color image processing literature will show the huge amount of research dedicated to basic primitives like color image segmentation. This results in a vast variety of approaches with different motivations and tools. The reader is referred to [246, 327] and the references therein. Though, no numerical criterion has ever been proposed for meaningful comparison of different segmentation approaches. After all, the only acceptable factors are the elapsed time and the visual properness of the results. Also, the utilization of many techniques include many parameters which should be tuned by an expert. Failing to do what will result in total or partial collapse of the method (see [97] as an example).

2.2.9 Alpha Estimation

Far from segmentation in usage, A similar concept called alpha–estimation is investigated in computer graphics [308, 261, 137, 135]. Need to Alpha–Estimation arises

when one tries to cut a natural object (like hair or leaves) from an image in order to paste it to another image [308]. In these cases no exact separation line can be find between the two objects. Many methods have been proposed for Alpha–Estimation, but they need the user to draw an estimation of the separating line, while the operations are all massively time–consuming [308, 261, 137, 135]. In this thesis, we do not enter alpha estimation in full details. It is only shown that the proposed fuzzy approach ro segmentation and region extraction may serve as an alpha estimator.

2.2.10 Compression

The early approach towards color image compression, is based on decorrelating the color planes using some linear or nonlinear invertible coordinate transformations (*e.g.*, YC_bC_r [5], YIQ [58], and YUV [328]), and then performing one of the standard gray–scale compression methods (like *differential pulse code modulation*(DPCM) [243] or transform coding [290]) on each plane, separately (see also [145]). This approach is inefficient, because none of the available color spaces are able to completely decorrelate the planes in a real image. In [104] using the PCA approach in the neighborhood pixels, the author discusses the idea of separating the spatial and spectral compression stages. As the paper proves, the maximum theoretical compression ratio for an ideal spectral compression method is $3:1$. The main shortcoming of the method in [104] is neglecting the fact that in non–homogenous regions, the PCA does not perform energy compaction [84]. In [84] the author combines the spatial and the spectral

information to reach a higher compression ratio. Although, the method is based on expensive computation, the PSNR results are desperate. The main shortcoming of the method in [84], is the block–wise artifacts produced after decompression.

2.2.11 Watermarking

Watermarking the artworks is attracting more attention because the possibility of electronic storage and transmission makes the probability of piracy extensively more [344]. Some researchers work on methods for general–purpose digital data watermarking (*e.g.*, [229, 399]). These approaches do not use the problem–specific constraints and possibilities. Also, as watermarking has a very tight relation to the cryptography, some researchers try to solve both problems at once (*e.g.*, see [229, 360]). Here, we separate these two different approaches of finding some perceptually non–important places in the image for embedding some data, and to encrypt that data.

Since the birth of the watermarking concept, many researchers have worked on the redundancy of images of the nature in the spatial domain, resulting in high performance spatial watermarking techniques (for a comprehensive survey look at [344]). Although, some researchers have worked on data hiding using the color redundancy (*e.g.*, see [399]), small work is performed on finding the real redundant color data in an image. Generally, researchers work on some subjectively–investigated assumptions about the color channels [399, 150, 101]. For example in [399] the researchers add the watermark to the blue plane and in [101] the authors use the $CIE - Lab$ color space

in a quantization framework to add the color watermark in the least significant bits.

2.2.12 Direct Copy Protection

The spread of digital media has enabled many kinds of fraud that harm the intellectual community. The availability of the Internet and the low cost of digital storage and copying devices has enabled data piracy to high extents that were even unimaginable a few decades ago. Thus, many researchers have worked on copyright protection methods [344].

Here, we address the problem of visual data protection. One of the main approaches in this field is watermarking [304]. In watermarking, a legal data is added to the original data to help identifying the original owner. Historically, during last decades different approaches for image (*e.g.*, see [229, 399, 101, 360]) and video (*e.g.*, see [103, 78, 63, 249, 232, 381, 255, 120, 228]) watermarking have been developed but none of them is claimed to be ultimately resistent to all kinds of attacks. Note that the meaningfulness of the attack–resistance, reported for some of the available methods, should be carefully regarded (for a full analysis of attack–resistance see [230, 284, 282] and *StirMark* [283]). In fact, development of cheating methods and the range of expert people taking part on them, makes it undoubtable that for any watermarking method, there is generally affordable methods of attack to produce non–watermarked images or even embed another watermark [369]. For example, see [159] for a set of guidelines for ultimate image copy protection from a graphics specialist in NASA,

containing words like "*never expose an image in its large size*", and suggesting to use "*visual watermarks*" and "*programming shield*".

Also, recently, the integrity of the watermarking methodology and its properness for security is essentially criticized [169]. In the afterwards comments on *Herley*'s controversial note "*Why Watermarking is Nonsense*" [169], researchers emphasized that this young field of signal processing is "*oversold*" and that no method has yet been able to claim "*the ability to protect from all possible future attacks*" [263, 10]. Researchers emphasize on nonsecurity–oriented application of watermarking and count on the new methods to come to help in the security field [10]. By the way, *M. Barni* makes the point clear "*Why should we hide the information within the data, when we could more easily use headers or other means to reach the same goal?*" [11]. In our opinion this forum is still open. See [108] for a survey of security scenarios in music watermarking and their failure. In a much different approach, a few researchers work on direct copy detection [210], but it is not yet an stable practical tool, too.

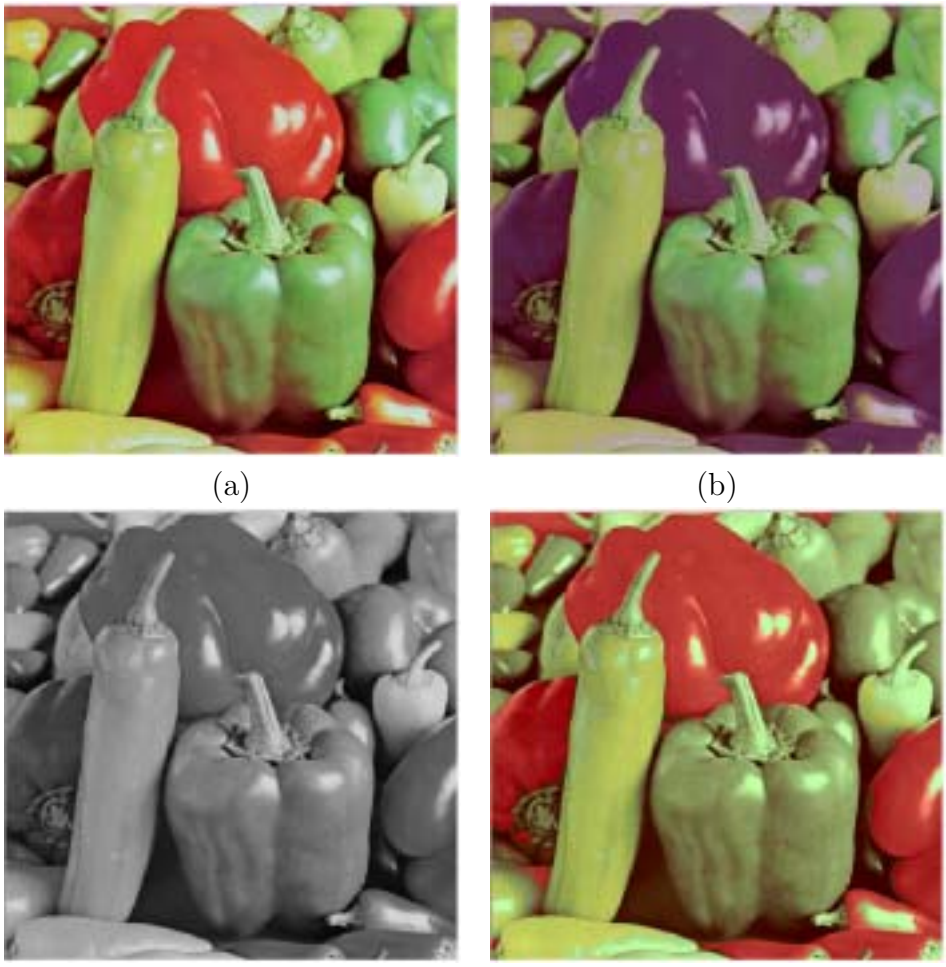
2.2.13 Color Transfer

Color transfer refers to the category of methods designed to change the color appearance of an image according to the color content of another image. The category of color transfer approaches contains two major categories of colorizing [375, 387, 367, 175, 238, 98, 153] and recoloring [297, 396, 92, 156]. When we wish to convert the grayscale source into a colored representation using the color information available

in the reference image, the method is called colorizing. Recoloring refers to the case where we wish to change the color appearance of a colored image to pretend another image's color content. Figure 2.33 shows typical results of colorizing and recoloring along with the original images. In the following, we address the available literature on both approaches of color transfer.

2.2.14 Colorizing

Although, many modern imaging systems are still producing grayscale images, color images are more preferred due to their larger information content. Computing the grayscale representation of a color image is a trivial task. However, when dealing with the inverse problem the process shows itself as a more complicated job; that should be performed with some levels of human intervention. Having in mind the enormous amount of grayscale classical heritage such as the first photograph, taken by *Lumiere* brothers, or the early movies of *Charles Chaplin*, the need for a user–friendly tool that converts the noncolored images into colored images gets clear. The commercial motivations towards such a colorizing process are not less than humanistic wills. Even nowadays, many imaging systems are still producing the old style grayscale images (*e.g.*, many types of medical images) or green–scale (*e.g.*, infrared imaging systems) images. To name a few application of colorizing we can mention the remote sensing images, medical images (for better illustration), surveillance images (for highlighting possible dangers; as the baggage inspection performed in the airports), and the



(a) (b)

(c) (d)

Figure 2.33: Results of the proposed color transfer methods. (a) and (b) Recoloring. (c) and (d) Colorizing. (a) Original color image. (b) Recolorized image. (c) Original grayscale image. (d) Colorized image.

weather images.

Colorizing refers to assigning a 3–D color vectors to a 1–D luminance or brightness realization. Thus, the problem is absolutely ill–posed with no exact solution. Figure 2.34 illustrates the flowchart of a typical color transfer process. It is worth to mention that different researchers use different terms to name the images taking part in this process. Here, we call the image from which the color information is extracted the "*reference*" image, while the image on which these color information is applied is called the "*source*" image. Finally, the source image with the color information added to it is called the "*destination*" image. In the system shown in Figure 2.34, a few color categories are extracted from the reference image. Also, a mechanism is designed to generate the corresponding color vector to a given grayscale value given that it belongs to one of the above mentioned color categories. Consequently, the heart of the system is the classifier that decides the associated color category for each pixel in the source image. Using this methodology, we will briefly review the available colorizing literature.

The simplest solution ever raised for the colorizing problem is the pseducoloring first introduced by *Gonzalez* and *Wintz* [153]. In their approach, no reference image is used and the color categories are 255 manually selected color samples. Also, there is no special classifier presented in their work; as the gray value g is assigned to the g–th color category, for which the color generator produces the g–th color sample

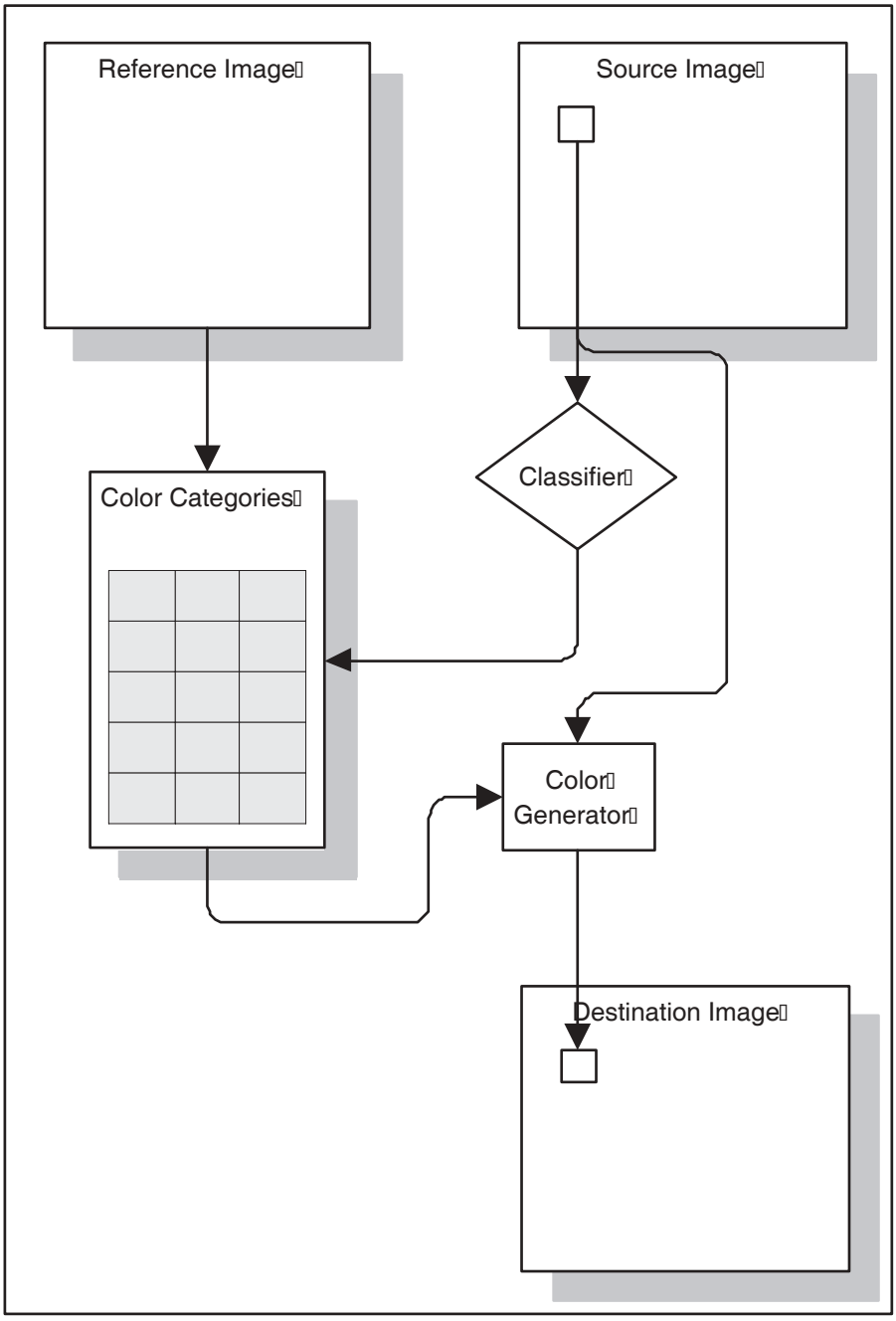


Figure 2.34: A typical color transfer system.

with no further manipulations. Although, the pseducoloring process seems to be fully automated, the process that generates the color map for the general problem, is yet another problem.

In an innovative work, *Welsh*, *Ashikhmin*, and *Mueller* [375] applied the $\ell\alpha\beta$ color space designed by *Ruderman*, *Cronin*, and *Chiao* [305] to the colorizing process. After converting both source and reference images to the $\ell\alpha\beta$ representations, their approach works in two modes. In the first mode, no color category is defined and the classifier searches each ℓ value from the source image in the entire reference image. To boost the results, the method also uses the statistics of the 5×5 neighborhood of each pixel in the comparison stage. This mode is only applicable for the images, in which "*corresponding color regions between the two images also correspond in luminance values*" as the paper mentions [375]. In the second mode, the method lets the user to define a set of corresponding swatches in the two images, and then the search is performed with more guidance. The color generator in their work, adds the α and β values to the pixel, while keeping the original ℓ value. Although, giving some pleasant results, the method is only applicable for the images divided into distinct luminance clusters and it fails for the images containing faces. The elapsed time by the procedure on a PIII 900 MHz PC using MATLAB code is reported to be in the range of 15 seconds to 4 minutes in an average image. Although, the paper concludes that the approach "*is not claimed to work on most images*" [375], the method is used

by many other researchers [387, 367, 98].

In a different approach, *Horiuchi* and *Hirano* [175], *Horiuchi* [174], and *Levin*, *Lischinski*, and *Weiss* [238], use optimization framework for colorizing. *Horiuchi et al.* [175] use a set of seed points and the corresponding color vectors, given by the user, and propagate the color to the entire pixels. Although, there is no reference image in this work, the seed points behave as the representatives of the color categories. The color generator works in the RGB color space and the classifier uses the Y component of the YUV color space. Roughly speaking, the method in [175], interpolates the RGB coordinates given that in each pixel the produced color vector must satisfy the initial Y value in the grayscale image. The paper reports the elapsed time as a few seconds [175]. In a more deliberated approach, *Levin et al.* [238] use quadratic objective function–based optimization task to interpolate the U and V components of the YUV color space over the entire image. In this work, rather than the grid of the seed points in [175] the user is allowed to define a set of color scribble lines, which are interpolated over the entire image. The method elapses 15 seconds for average–sized images.

There is a literature for colorizing cartoon images [345, 346], that assumes that the background is a static image and that all objects consist of several well visible outlines that emphasize the shape of homogenous regions. This approach completely differs from colorizing real photographs and paintings and thus is not discussed here.

2.2.15 Recoloring

Each image has its own color feeling that greatly influences the perception of human observers. Being able to transfer the color content of an image into another image, while preserving other features like the texture, opens a new horizon in human perception based color image processing.

Reinhard, *Ashikhmin*, *Gooch*, and *Shirley* [297] are perhaps the first team to work on color transfer. They describe their core strategy as choosing a suitable color space for the color generator which works in an statistical framework. Using the $\ell\alpha\beta$ color space for the sake of decorrelating the color components, their first approach for recoloring is to map the pixels from the stochastic distribution of the source image to the reference image as,

$$x^{\star} = \frac{\sigma'_x}{\sigma_x}(x - \eta_x) + \eta'_x, \tag{2.2.27}$$

where x is one the color components of ℓ, α, or β. Also, σ_x, σ'_x, η_x, and η'_x refer to the standard deviation of the color component x in the source and the reference image and the standard deviation of the two images, respectively. In that scheme, the whole image is defined as a single color category, so there is no color classifier present. The color generator is a linear mapping function. As the paper describes, the performance of the method depends on the similarity of the images; for example, if the source image contains more grass and the reference image has more sky in it, then we can expect the transfer of the statistics to fail [297]. To overcome this

shortcoming, *Reinhard et al.* propose to use the swatches. In that method, user must select two sets of corresponding swatches in the source and the reference images, for example the grass, the sky, and the forth. Then each swatch is defined as a single color category. For the given color vector, the classifier computes its fuzzy membership to each color category using the inverse Euclidean distance. The color generator uses equation (2.2.27) in each class to find the new color vector, given that the input pixel belongs completely to that class. Then, the resulting vectors are blent using the membership values. One of the main contributions of the work is the idea of computing the altered color vectors in each class and then merging the results. They report promising results and suggest further use of higher moments.

Chang, *Saito*, and *Nakajima* [92] state that as each artist has its own style like the "*yellow that Van Gogh likes to use*", it will be helpful to be able to transfer the feeling of a painting to a photograph, which is what they call "*giving the viewers the impression similar to that given by the painting*". Their work is entirely devoted to color transfer from a color painting to a color photograph. *Chang et al.* [92] use an early work by *Berlin* and *Kay* [56]. Where, they examine 98 languages from several families and report that there are regularities in the number of basic colors and their spread on the color space. *Chang et al.*'s work [92], that is implemented entirely in the $CIE-Lab$ color space, uses later works that define the spread of these categories. The method begins with making the 11 loci of the points in the source image that

belong to any of the 11 clusters. Then, they generate the convex hull that encloses all the pixels within each of the categories. The same task is performed in the reference image. These 11 loci make the color categories and the classification task investigates the locus inside which the given color vector exists. The color generator uses geometric mapping using the parameters of the locus to which the given color vector belongs and the corresponding locus in the reference image. As *Chang et al.* [92] use a set of fixed swatches, the method does not let any user supervision. Also, as the color transfer is limited to be performed within each fixed color category, the user is not allowed for example to add reddish reflections to the blue surface of the sea like what occurs due to the sun–shades. This manipulation is obtainable using the method by *Reinhard et al.* [297].

Yin, *Jia*, and *Morrissey* [396] developed a color transfer method especially designed for face images. Thus, there is no color classifier presented in their work. Their proposed color generator is the same as *Reinhard et al.*'s [297] but works in the HSI color space.

Greenfield and *House* designed a method for color transfer between color paintings using the $\ell\alpha\beta$ color space [156]. They organized the source and the reference images into a pyramid to produce a palette for each image by successively down–sampling it. *Greenfield et al.* [156] emphasize that their primary focus is not making an intelligent pallet association process. Then, using some heuristic methods counting on

the area occupied by each pallet color they propose a simple association method. We emphasize that this is far from a practical solution. Each pallet color serves as a color category and the classifier works in a hierarchical way to assign a pallet color to the given pixel. The color generator transfers the α and β components and then performs a color correction process to compensate for ℓ variations. While the method by *Greenfield et al.* [156] suffers from the same problem of the *Chang et al.*'s approach [92] (because of leaving no room for user intervention) they use the spatial distribution of the color vectors more professionally. One of the main shortcoming of the method by *Chang et al.* [92] is the assumptions for pallet association (like the decision that the material occupying the largest amount of the source image also covers the largest area of the reference image). They have no comment about the performance of their approach for color photographs.

Remark

Neglecting the details, all of the available color transfer methods (containing both colorization and recoloring approaches) use the same assumption that there exists a color space that performs well in decorrelating the color components. It is proved in different works (*e.g.*, [393, 323, 191]) that none of the available color spaces are successful in giving a decorrelated representation of the color vectors in an unconditioned imaging framework for classification or the kinds of direct manipulations mentioned here. Also, the Euclidean distance–based classifiers used by *Reinhard et al.* [297] and

Welsh et al. [375] is proved to result spuriously (see Section 4.3.

2.2.16 Content–based Pornography Filtering

With the rapid penetration of the Internet into every part of our daily life, it is agreed that it will be an important media for future communication, perhaps even more important than the television [119]. In 1996, it was estimated that the Internet was grown into an international network, connecting over 60 million users in more that 60 countries [77].

Though, it is proved, even for non–regular users of the Internet, that it is getting more and more contaminated with harmful content, such as pornography, violence, hatred and so on. In a recent survey, one in four kids reported having at least one unwanted exposure to sexually explicit pictures during the past year, and one out of five reported receiving a sexual solicitation [3]. Also, an early investigation by a university student showed that approximately 83% of the images in the Usenet groups were pornographic (in 1995) [130]. Recent statistics reveal that 25% of the queries in search engines, 8% of emails, and 12% of homepages are porn–related [8, 9]. Also, the average age of first Internet exposure to pornography is 11 years old [8]. See [1] for more statistics.

The content analysis of pornographic images available in the Internet in [258] suggests that commercial vendors post explicit pornographic material in public access newsgroups to attract new customers to their private, pay–per–use bulletin board

services. A good investigation of the porno–market can be found at [106]. Also, see [349] for a psychological–legal survey and [227] for a criminological discussion.

While parents and educators tend to have more control on the information exposed to their children when surfing the web [116], the common textual approach (such as *NetNanny*, *CyberSitter*, and *SurfWatch*) is not performing well [157]. While failing to give enough immunity, the text–filtering approach blocks many useful sites, just because of the presence of some certain phrases. For example, a commercial package blocked the White House children's page in 1996 [6]. On the other hand, the owners of the harmful sites are inventing new methods for shutting down the text–searching immunity system. Approaches like only–picture pages and safely–named files are now just common. In recent years, the pictorial content–based filtering is gathering more and more attention.

To have an idea about the importance of the images for content–based Internet filtering, it is helpful to review the results of a recent survey. The statistics of more than 4,000,000 HTML pages reveal that more than 70% of them contain images [338]. Also, the results show that a single HTML file, averagely contains 18.8 images [338]. The statistical review of $1,232$ pornographic and $6,967$ non–pornographic webpages shows that 72% of pornographic webpages have more than 5 images and 60% of them have more than 10 images [171]. Also, 40% of pornographic webpages have more than 5 links to image and video files. Hence, the ability to detect pornographic images is

a helpful tool towards filtering adult pages in the Internet.

One should be aware that the exact definition of an erotic image differs in different countries. For examples in most western cultures, homepages dedicated to the sales of underwear or swim wear are not considered to be pornographic. Though, in some eastern countries they do. The exact definition of pornography is even undefined in less–conservative countries. For example, in UK there are seven acts of parliament concerning pornography [7], but in US, the *Communications Decency Act* was ruled unconstitutional in June 1997 and its successor, the *Child Protection Act* is subject to legal challenge. This subject is out of the scope of this thesis and the interested reader is referred to [299, 126, 160, 358]. In this way, we agree with the authors in [329] that "*pornography is an ill–defined concept and that we use it loosely, to refer to unwanted material with a sexual content*".

Figure 2.35 shows the number of published scientific papers (either in conferences or journals) devoted to content–based pornography detection. The chart covers the period of 1996 to 2004, because the first known works are published in 1996 [136, 141]. This figure depicts the motivation of scientific society to solve the problem of unwanted exposure of pornography in the Internet.

The first work on classifying nude pictures belongs to *Forsyth et. al.* [136, 141, 140]. In that approach, firstly, the images containing large areas of skin are detected (using IR_gB_y color space) and then using a specially defined human structure [185, 184], the

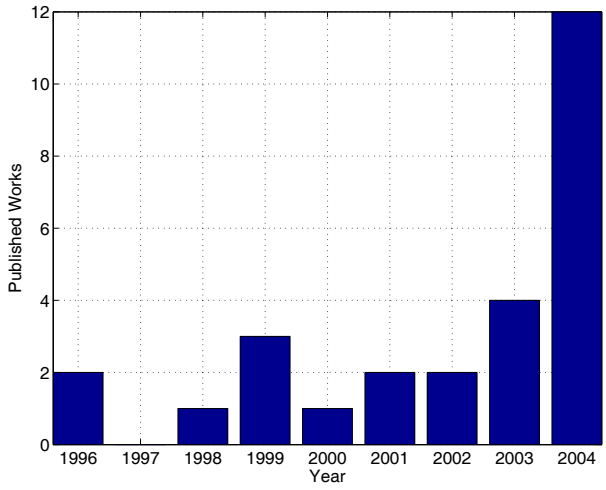


Figure 2.35: Number of scientific publications dedicated to content–based pornography filtering in the years of 1996 to 2004.

images containing human figures are recognized as the nude ones. In their approach, only the images which are at least covered in one third by the skin area are fed to a geometric analyzer. While all the images used in [136, 141, 140] are 128×192, it takes about six minutes for the method to process a single image. Though, the final recall of the method is less than 50%.

After that, other researchers made new contributions in this field. In [371], the researchers utilize skin detection in *RGB* color space equipped with wavelets and central moments on 128×128 images to develop the WIPE^{TM} system.

In [329, 91, 90], the researchers combine the results of a simple skin area estimator based on a skin detection filter with textual analysis results. That paper reports that "*while a threshold is selected, the choice of the color space is not critical. This is*

somewhat surprising because other authors advocate the use of IR_gB_y". That work is extended in [67] using a specific neural network to raise the recall rate up to 90%. The researchers draw the final conclusion that "*provided that enough training data and a histogram–based representation of the color distribution are used, then the choice of color space is not critical*" [67].

In [195], the authors describe the steps towards developing an adult image detection scheme using the skin color in YUV, YIQ, and their polar representations (YUV°, YIQ°). In that approach, a Sobel edge detector and a Gabor filter judge the results of the skin detector. In [122] the researchers use $CIE - Lab$ color space for detecting the skin areas in the images. The results are then analyzed using some geometrical features.

In [83], the researchers compute some shape descriptors using the results of skin tone detection in YC_bC_r and IR_gB_y. Then, using an Adaboost learning scheme, the erotic images are classified. In [199], the researchers combine the results of a neural network classifier working on image features with a commercial text–filtering tool developed by *SurfWatch* (used by *Altavista*). The image feature used in that work only relates to the portion of pixels identified to be skin (in RGB color space), with no further geometric processing.

The skin detector in [198] is used in [398, 397] to develop a MPEG7–based adult image identification system, using query–based search into an erotic dataset. In the

approach described in [316] the results of a skin detector in RGB and HSV color spaces are used to produce other features like a simple texture descriptor. Then, all of the features are fed to a *state vector machine* (SVM). In [244], the researchers use the results of a straight skin–detection rule ($R > G$ or $R > B$) with a SVM classifier. In [163, 164] the researchers give general cues for developing a web–filtering tool.

In [194], a gaussian mixture model is equipped with Baysian inference to utilize the skin detection stage. Then, the results are fed to a SVM. In [54], the researchers utilize genetic algorithm for locating the loci of nude pictures in a search space containing huge 801–D feature vectors. Each feature vector contains color histograms in the C_b and C_r color components plus some shape descriptors. Alongside using the color components' histograms in the feature vector, the method in [54] devises hard thresholds on the C_b and C_r components to extract the skin–like objects ($78 \leq C_b \leq 135, 85 \leq C_r \leq 185$). In [411, 409], the researchers use the results of a skin detector in the RGB color space for a multi–layer perceptron classifier. In extended works they applied a better skin detection filter developed in [198] (see also [407, 408, 410, 406]). In [196] researchers utilize a custom non–linear color space. In [412] a multi–Baysian skin detector is utilized.

The information about the available pornography filtering approaches are listed in Table 2.5.

Table 2.5: Summary of the available pornography detection works.

Work	Color Space	Approach
[141, 140, 136]	IR_gB_y	OLUT
[371]	RGB	OLUT
[329, 91, 67, 90]	RGB, HSV, IR_gB_y, $Nrgb$	BLUT
[195]	YUV, YIQ, YUV°,YIQ°	OLUT
[122]	$CIE-Lab$	Not Available
[83]	YC_bC_r, IR_gB_y	OLUT
[199]	RGB	OLUT
[398, 397]	RGB	Histogram–Based
[316]	RGB,HSV	OLUT
[244]	RGB,HSV,IR_gB_y,$NRGB$,YC_bC_r	OLUT
[194]	RGB	BLUT
[54]	C_bC_r	OLUT
[411, 409, 407] [408, 410, 406]	RGB	OLUT
[196]	Custom	BLUT
[412]	RGB, RGB^r	BLUT

2.2.17 Skin Detection

The review in Section 2.2.16 shows that it is generally accepted that the first step towards finding the whole human body or its selected parts is to find those parts of the image holding pixels corresponding to the skin material. This assumption is theoretically backed by models of the skin tissue. In this way, it is argued that the color of human's skin is created by a combination of blood (red) and melanin (yellow, brown) [302]. Most researchers presume from these evidence that the skin color can be recognized in an image with no explicit knowledge about the lighting conditions, the camera calibration, and the subject properties such as the race.

A skin detection system consists of two major parts. First, there should be a

proper model for representing colors, what we call the color space. Second, there is an inference methodology to obtain information from available skin samples and to extrapolate the results to given samples. For the first part, Table 2.6 shows that the solution of different researchers seems to be more related to personal taste rather than experimental evidence [393]. We will discuss this point with more details in this thesis.

While there are plenty of methods for modelling and detecting skin tone, as Table 2.5 shows, in pornography detection field, researchers tend to gather around two major choices. From the twenty–six available approaches for pornography detection, sixteen use ordinary look–up table, seven use Baysian look–up table, and only three utilize a different approach for skin detection. Hence, here we focus on the two methods for look–up–table–based skin detection. As such, the concentration of this thesis is on the best choice for representing color vectors.

Few works are available that compare the performance of different color spaces for skin detection [323, 191, 152, 79, 405, 402, 352, 69, 49], but they are all concerned with face detection. As mentioned in [368] "*pixel–based skin detection has long history, but surprisingly few papers that provide surveys ... were published*". for a survey of the available color space comparison works see [368]. We argue that as the conditions for pornography filtering are essentially different from other skin–related fields that allow more–or–less constraints on the imaging conditions, a more precise and focused

Table 2.6: Diversity of color spaces utilized for skin detection.

Work	**Color Space**
[198, 200, 69, 70, 314, 192, 193] [339, 91, 294, 264, 286, 90, 67] [329, 412]	RGB
[339, 62, 201, 325, 402, 404, 403] [257, 333, 331, 372, 170, 218, 72] [91, 67, 152, 405, 315, 244, 352] [155, 132, 90, 329]	HSI, HSV, HSI^c, HSV^c, HLS
[94, 352, 382, 96]	$CIE - XYZ$
[402, 404, 403, 405, 259, 48, 85] [50, 336, 373, 86, 339, 244, 152] [88, 285, 179, 79, 315, 392]	YC_bC_r
[402, 404, 403, 405, 391, 317, 392] [152, 81, 352, 122]	$CIE - Lab$
[113, 114, 339, 69, 370, 195, 152] [252, 195, 105, 352, 310, 309, 385]	YIQ, YIQ° YUV, YUV°
[392, 402, 404, 403, 405, 391, 317] [152, 392, 352]	$CIE - Luv$
[140, 141, 185, 184, 136, 244, 402] [405, 67, 91, 329, 90]	IR_gB_Y, $IR_gB_Y^+$
[131, 216, 217, 342, 292, 262, 274] [109, 110, 390, 389, 392, 337, 321] [244, 72, 341, 181, 339, 402, 405] [352, 155, 129, 67, 91, 329, 90]	$Nrgb$
[69, 374, 152, 79, 412]	RGB^r
[352, 72, 79]	TSL
[351]	Others

investigation in this field is necessary.

The contributions of this thesis are in three ways. Firstly, here we use all samples and training data from pornography resources. In this way, we concentrate on the best choice of color space for pornography detection. In this way, this research is dedicated to detecting skin tone in pornography images taken in absolutely arbitrary conditions. Secondly, we arrange the largest number of color spaces ever used in an evaluation. While the available works compare less than ten color spaces with each other, we gather 21 color spaces in a unique experiment. The third contribution of this work is that we make no presumptions about the best dimension of color spaces. All available works use the original 3–D color spaces, or cut them into 2–D representations by neglecting the component assumed to be related to illumination. In contrast, we perform all the experiments on each color space in all its possible 3–, 2–, and 1–D representations. Also, more than computing the performance of different color models for the training and test data, we perform a real skin detection task for a large erotic dataset and evaluate different color spaces in terms of corresponding results.

2.3 Color Image Sequence Processing

The need for handling the ever increasing image and video databases has emerged the usefulness of fast and reliable multimedia management systems [272, 330]. Among the basis low–level image descriptors, color is proved to be one of the most effective

feature; because it is independent to the geometrical changes [280].

2.3.1 Segmentation

Spectral–spatio–temporal segmentation is the process of finding spatially connected regions that represent spectrally homogenous fields in image sequences. Such method should result in a (possibly fuzzy) partitioning of the image into its constructing elements [278].

One of the fundamental approaches of grayscale video segmentation is based on estimation of the motion vectors by tracking the edges, regions, and the forth [365]. This approach is massively time consuming [213]. For example, the method described in [278] needs one minute for processing one frame of a sequence with two moving objects, and when working on three moving objects elapses three minutes, using a PII 300MHz personal computer and 512×512 frames. In [362] the authors use region labeling while detecting scene–cuts using the mean square error between the corresponding pixel's grayscale values in consecutive frames. When a scene–cut is detected, all segmentation information is ignored and the algorithm restarts the motion estimation procedure. Working on a PIII 500MHz personal computer, the method elapses only 2 seconds on each frame of a sequence with stationary background and $5s$ to $10s$ on each frame of a sequence with moving background (512×512 frames). In [278] the authors down–sample the image pixels to gain reasonable performance. In [214] a self organized feature map approach for video segmentation is introduced

that elapses more than 6 minutes on each frame. The method uses many parameters left to be tuned by the user. Although, the method in [245] gives acceptable results, it depends on 11 parameters. This high dependency on parameter–tuning makes the method improper for many applications. Also there are naive applications of morphological filtering for video segmentation giving fairly acceptable results in highly controlled environments like distance learning [211].

The literature of color video segmentation is not so rich compared to the case of grayscale segmentation methods. The first significant method for measuring the color–based similarity between two images might be the color histogram intersection approach by *Swain* and *Ballard* [343]. Although, the method is very simple, it gives a relatively reasonable performance with two main shortcomings: the lack of spatial information about the images, and dependency to imaging conditions (like the ambient illumination). Some other researchers try to use certain color spaces that they believe them to be suitable for segmentation. For example in [234] the authors use a geometrical measure in the color histogram to define the similarity between color pairs in the HLS color space. Although, some good segmentation results in the HLS color space are reported [53], it is proved in various studies that none of the available color spaces are outperforming the others (*e.g.* see [323]), while the local principal component analysis is proved to give dominantly better results (see Section 4.3). In [123], the researchers process color components independently, neglecting the vector tendency

of them. In [281] motion estimation is used for segmentation purposes.

2.3.2 Change Detection

Change Detection in Image Sequences

Change detection is the process of finding the locations of changes between two successive frames. Change detection is a crucial primitive stage in video analysis, object tracking, and computer vision. Also, many MPEG video standards rely on change detection as an early stage in video object extraction process. The possibility of robust change detection has even resulted in some novel applications in image sequence processing [384].

Although, change detection is an active research area, researchers tend to consider change detection as a tool working on grayscale images, finding crisp change detection results of either changed or unchanged regions [324]. Majority of works in this field use the subtraction of successive frames either by pixel–wise [117] or block manners [376]. Different researchers apply different methods on these frame–to–frame differences. Mixture of Gaussian or Laplacian distributions [364], Kalman filtering [204], Markov 1–D chains [80], Markov random fields [271], contour evaluation [206], region growing [158], and adaptive thresholding [215] are only a few examples of more than two decades research on change detection. Also, there are many methods that give acceptable results but depends on a handful of parameters to be set for each sample by a human observer during trial and error steps (*e.g.*, see [413, 65]). However, all above

mentioned methods are working on grayscale images to make crisp change detection decision for each pixel.

There is an absolutely narrower literature for change detection in color images. Though, many of the available works on color change detection use the color information to enhance the results (and the grayscale information is ruling the method (*e.g.*, see [241])).

Change Detection in Remote Sensing Images

Change detection algorithms usually take two images as the two shots before and after the change and return the locations where the change is likely to be happened [379]. Here, we assume that the given images are registered in the preprocess stage.

A fundamental problem in comparing two shots of the same scene in different times, is the changing recording conditions. As such, a change detection algorithms needs to be robust against atmospheric and imaging parameters. In particular, the direct solar illumination, the diffuse sky light, the path radiance, and the transmittance of the atmosphere, as well as the dark current and gain setting of the sensor may have changed individually in each spectra band [379]. All these effects can roughly be categorized into multiplicative and additive influences on the acquired data. Thus, the relationship between two shots of the same scene in different times is very often modelled approximately as a linear function [326, 298]. Many authors use spectral pre–processing to compensate these changes in the imaging parameters using ratioing

and differentiating approach [59].

There is a rich literature of change detection in gray–scale images. In [296] the authors survey the main approaches and compare the results. Some more frequently used methods in this field are correlation-based change detection [59], Baysian thresholding [75, 76], Artificial Neural Networks [111], and multi–block PCA [293], to name a few. Also, some researchers have tried object–based change detection, trying to locate the buildings and using morphological tools to estimate the damaged locations [51].

Compared to the single–spectrum change detection, multi–spectral change detection is being worked much less. The most frequent approach in this field is using the PCA to find the relation between the two images and then mark those not complying with this relation as the locations of the change [298, 379]. Some other researchers are using regression to find the non–changed axis [386]. Also, the application of clustering–based methods [359, 253], special color spaces [388], and the linear discriminants [82] are reported. Another idea for finding the locations of change in multispectral images is to use the exact signature of the constructing material of the scene [260]. This approach is based on expensive accurate laboratory experiments.

2.3.3 Recoloring

Figure 2.36 shows a sample result of the proposed recoloring method applied on an still image. Due to the complexity of the available color transfer methods, no attempt

(a) (b) (c)

Figure 2.36: Sample result of the proposed recoloring method on still images. (a) Source image. (b) Reference image. (c) Recolored image.

is made in utilizing them for processing color image sequences.

2.4 Registration

In recent years, the spatial and the spectral resolution of the remotely sensed sensors and the revisiting frequency of the satellites, is extensively increased. These developments, has offered the possibility of addressing new applications of remote–sensing in environmental monitoring. On the other hand, the officials are getting more and more aware of using multi–spectral remote–sensing images for regular and efficient control of the environment [379, 134].

A key issue in remote–sensing image analysis is to detect changes on the earth surface, in order to manage possible interventions to avoid massive environmental problems [59]. Recently, many researchers have worked on using the remote–sensing

data to help estimating the earthquake damages [256, 51] or the afterwards reconstruction progresses [357, 265].

The process of registration aims at performing some geometrical operations on one of the images, (or both of them), to give two compatible images; in which the pixels with the same coordinates in the two images correspond to the same physical point [73]. Many researchers have reported the impact of miss–registration on the change detection results (*e.g.*, see [356]). The registration operation is an inverse problem, trying to compensate the real transformation produced by the imaging conditions. Although, different registration methods are introduced and analyzed [73, 414], there is no optimal solution yet found and the problem is still an active research area [300].

The majority of registration methods consist of four essential steps [414]:

1. Feature detection.
2. Feature matching.
3. Transfer model estimation.
4. Image resampling and transformation.

The first step along with the second step aim at finding two sets of corresponding points in the two images. These two sets are used in the second step to find the actual transformation. Finally, the fourth step gives the two registered images.

There are two typical methods for finding and matching feature points. The first one is to search for salient points in the two images. There are reports of using contours [240], boundaries [383], water reservoirs [172, 154], buildings [178], urban areas [303], roads [242], forests [319], costal line [250] and the forth as the features. Another approach is to use the information theory tools like mutual information to find the control points [266]. All of the above mentioned approaches perform both feature detection and feature matching at the same time. Due to the massive effect of mismatching of the control points on the final registration result [356], we criticize the applicability of the automatic approaches in this application. Note that, using registration for change detection of remotely–sensed images differs from application in video surveillance because of the usage frequency. In fact, remote sensing images are still very expensive and accuracy is critical for them. Thus, we propose to use the old–style approach of using human intervention for finding a set of about 20 control points in the two images. The non–efficiency of automatic approaches is more clear when investigating the post–earthquake images (see Figure 2.37). It must be emphasized that any automatic control point detection method can be integrated to the proposed method.

2.5 Principal Component Analysis (PCA)

The *principle component analysis* (*PCA*) [209, 197, 183] is widely used in signal processing, statistics, and neural networks. In some areas, it is called the (discrete)

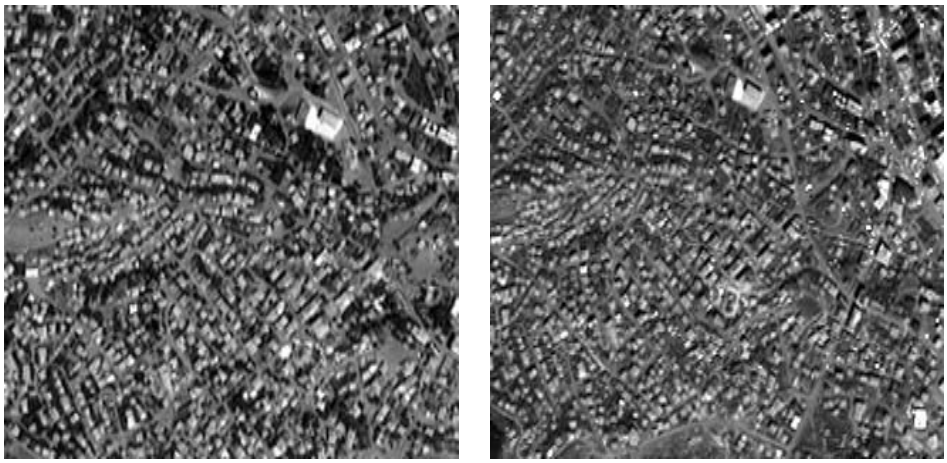


Figure 2.37: *Bingol*, *Turkey*, (©Digital Globe). (a) Before the earthquake 2002–07–15. (b) After the earthquake 2003–05–02.

Karhunen–Leove transform (in continuous case) or the Hotelling transform (in discrete case). The basic idea behind the PCA is to find the components $s_1 \dots s_n$, so that they explain the maximum amount of variance possible by n linearly transformed components. By defining the direction of the first principal component, say w_1, by (2.5.28), the PCA can be represented in an intuitive way [183].

$$\vec{v}_1 = \arg\left(\max_{|\vec{v}|=1}\left[E\{(\vec{v}^T x)^2\}\right]\right). \tag{2.5.28}$$

Thus, PCA is the projection of the data on the direction in which the variance of the projection is maximized. Having determined the first $k-1$ principal components, w_k is determined as the principal component of the residual stated as [183]:

$$\vec{v}_k = \arg\left(\max_{|\vec{v}|=1}\left[E\{(\vec{v}^T \Delta_{k-1})^2\}\right]\right), \tag{2.5.29}$$

Where Δ_{k-1} in (2.5.29) is defined as:

$$\Delta_{k-1} = x - \sum_{i=1}^{k-1} \vec{v_i}^T x \vec{v_i}. \tag{2.5.30}$$

The principal components are then given by $s_i = \vec{v_i}^T x$ [183].

In practice, the computation of w_i can be accomplished by using covariance matrix $C = E\{(x - \bar{x})(x - \bar{x})^T\}$. The w_n is the eigenvector of C corresponding to the n–th largest eigenvalue [183]. The basic goal in PCA is to reduce the data dimension. Thus, one usually chooses $n \ll m$. Indeed, it can be proven that the representation given by PCA is an optimal linear dimension reduction technique in the mean–square sense. Such a reduction in dimension has important benefits. First, the computational load of the subsequent processing stage is reduced. Second, noise can be reduced, as the data not contained in the n first components may be mostly due to noise [183].

Figure 2.38 shows the help section of the *pca* function included in **CIP**. **CIP** also includes an implementation of *pcamask* which performs the PCA for the points highlighted in a given binary mask (see Figure 2.39). The *pca* function inputs a color image, the value of p in the $[0, 100]$ range, and the value of kind as a member of the set $\{-2, -1, 0, 1, 2, 3\}$. Here, I is color image on which PCA will work. The value of p which is called the inclusion percentage, indicates the margin on the PDF of destination values. This margin will be reported as the non–homogeneity of I under regarding distance function. See Section 3.3 for further discussions. The value of kind decides which homogeneity values will be computed. When using PCA in an

approach not relevant to homogeneity decision, kind may be left -2. The outputs of PCA are the expectation vector E, the principal directions *v1*, *v2*, *v3*, and the three homogeneity values of *ere* (Mahalanobis), *err* (LPRE), *erq* (Euclidean). Also, PCA gives the V and D matrices which are the PCA matrix and the eigenvalue matrices. *err2* is the L_2 LPRE homogeneity value and *lambda* if the marginal size of the LPRE cylinder. The later one is used for the LLPRE measure discussed in Section 3.4.2. The distance functions are precisely discussed in Section 3.3. *pcamask* is similar to *pca* except for the input M which is a binary mask. As such, *pcamask* works on the set of pixels denoted by M.

PCA

```
function [E,v1,v2,v3,ere,err,erq,V,D,err2,lambda]=PCA(I,P,kind)
%PCA computes principle components of the image I.
%
%Usage:[E,v1,v2,v3,ere,err,erq,V,D,err2]=PCA(I,P,kind)
%Inputs: I: Input image in RGB, must be of type Uint8 .
%          If using imread, be sure to load a 24bit bmp file
%          Never use indexed images for I.
%          P:Inclusion percent multiplied by 100. (-1 forces to use mean)
%          kind: -2..3 Compute only the desired margin.
%                (0:Euclidean, 1:Mahalanobis, 2:Reconstruction Error,
%                  3:L_2 LPRE, -1:Compute all, -2:None.)
%Outputs: E: Expectation value for the three channels. 3*1 of type double
%           in range [0,255]
%         v1:Most important Principle components in 3*3 double format.
%         v2:Medium important principle component
%         v3:Least important principle component
%         ere:Average Mahalonobis distance between region points and the
%             region data cloud.
%         err:Average reconstruction error.
%         erq:Marginal Euclidean error.
%         V :Principle components. V(:,i) gives the i-th principle
%         component
%         D:Eigen values of the covariance matrix
%         err2:L_2-based LPRE
%         lambda:Marginal size of the cylinder for the LLPRE.
%
%PCA stands for Principle Component Analysis
%Revision 3.51, Arash Abadpour, Fall 2004 (Oct. 24 2004).
```

Figure 2.38: Implementation of the *pca* function.

PCAMask

```
function [E,v1,v2,v3,ere,err,erq,V,D]=PCAMask(I,M,P,kind)
%PCA mask computes principle components of the image I where highlighted by M.
%
%Usage:[E,v1,v2,v3,ere,err,erq,V,D]=PCAMask(I,M,P,kind)
%Inputs: I: Input image in RGB, must be of type Uint8 .
%          If using imread, be sure to load a 24bit bmp file
%          Never use indexed images for I.
%        M: Mask image (0,1)
%        P:Inclusion percent multiplied by 100. (-1 forces to use mean)
%          kind: -2..3 Compute only the desired margin.
%                (0:Euclidean, 1:Mahalanobis, 2:Reconstruction Error,
%                 3:L_2 LPRE, -1:Compute all, -2:None.)
%Outputs: E: Expectation value for the three channels. 3*1 of type double
%          in range [0,255].
%        v1:Most important Principle components in 3*3 double format.
%        v2:Medium important principle component.
%        v3:Least important principle component.
%        ere:Average Mahalonobis distance between region points and the
%            region data cloud.
%        err:Marginal reconstruction error.
%        erq:Marginal Euqlidean error.
%        V :Principle components. V(:,i) gives the i-th principle
%        component
%        D:Eigen values of the convariance matrix
%
%PCA stands for Principle Component Analysis
%Revision 2.3 (c) Arash Abadpour Fall 2004 (Oct. 2, 2004)
```

Figure 2.39: Implementation of the *pcamask* function.

Chapter 3

Proposed Methods

This section introduces the methods proposed in this thesis. Most of the algorithms discussed here are published in conferences and journals. Table 3.1 lists the published papers and corresponding sections in this thesis.

3.1 Mathematics

In this section, some mathematical concepts developed in this thesis are discussed. The discussion includes introducing *fuzzy principal component analysis* (FPCA) in Section 3.1.1, the special problem of weighted powered sum minimization in Section 3.1.2 which is used in the general fuzzy clustering algorithm in Section 3.1.3 and the basis vector polarization concept introduced in Section 3.1.4. Also, this section talks about block–wise interpolation (Section 3.1.5), butterworth membership function (Section 3.1.6), and the eigenimage concept (Section 3.1.7). This section then includes a brief discussion about the way to produce a uniform quantized random variable in Section 3.1.8 and introducing the quality measures used in this thesis in

Table 3.1: Papers published from this thesis and the corresponding sections.

Citation	**Section**
[21, 14]	3.2
[25]	3.3
[22]	3.6.1
[24]	3.9.1
[15, 23]	3.1.4, 3.1.5, 3.1.7, 3.7.1
[18]	3.1.6, 3.9.2
[17, 20]	3.1.1, 3.1.3, 3.5.2
[45]	3.5.3
[35, 36]	3.12
[31, 19]	3.4.1
[33]	3.5.1
[29]	3.10
[32, 28]	3.7.2
[43]	3.9.3
[34, 15, 44]	3.6.2
[26]	3.11.1
[40, 41]	3.13
[27]	3.1.8, 3.8.2
[16]	3.9.1, 3.9.2
[42]	3.8.1
[30]	D
[39]	3.1.11, 3.5.4
[37]	3.1.10, 3.11.3
Unpublished	3.4.2, 3.11.2

Section 3.1.9. The next parts of this section are devoted to finding the closest point to a set of lines in R^n in Section 3.1.10 and to a set of fuzzy lines in Section 3.1.11.

3.1.1 Fuzzy Principal Component Analysis (FPCA)

Consider performing the PCA transform on the set of vectors $X = \{\vec{x}_i | i = 1, \cdots, N\}$, while the samples are members of the discrete field $\Phi = \{\vec{\phi}_i | i = 1, \cdots, n\}$. This situation happens when we work on color vectors, which are repetitions of the vectors available in $\Phi = (N_{255} \cup \{0\})^3$. Hence, we are facing the PCA problem for the set of fuzzy vectors $\{(\vec{\phi}_i; p_i) | i = 1, \cdots, n\}$, where p_i is the number of repetitions of $\vec{\phi}_i$ (the histogram). Consider the general problem of finding the principal components of $\{(\vec{x}_i; p_i) | i = 1, \cdots, n\}$, when not restricting $\sum_{i=1}^{n} p_i$ to be 1. Here, as we are only concerned with the zero and one dimensional representations of the data cloud, we derive the formulation for $\vec{\eta}$ and $\vec{v}_1$, but the same method leads to the computation of the other fuzzy principal directions as well.

All objective functions in the PCA theory are of the form,

$$\Delta(\vec{x}_\circ) = \sum_{i=1}^{N} Q_{\vec{x}_\circ}(\vec{x}_i), \tag{3.1.1}$$

where, $Q_{\vec{x}_\circ}(\vec{x}_i)$ is a quadratic function of $\vec{x}_i$. Assuming the case of repeated vectors, the objective function changes to,

$$\Delta(\vec{x}_\circ) = \sum_{i=1}^{n} p_i Q_{\vec{x}_\circ}(\vec{\phi}_i), \tag{3.1.2}$$

where, p_i is the number of occurrences of $\vec{\phi}_i$. Thus, it is reasonable to define the

objective functions for the fuzzy set $\tilde{X} = \{(\vec{x}_i; p_i) | i = 1, \cdots, n\}$ in the fuzzy domain as,

$$\Delta(\vec{x}_\circ) = \sum_{i=1}^{n} p_i Q_{\vec{x}_\circ}(\vec{x}_i). \tag{3.1.3}$$

Assuming that all p_i are constant, (3.1.3) leads to the non–fuzzy definition. Also, the assumption of $\sum_{i=1}^{n} p_i = 1$ is not necessary, because it shows itself as a constant scaling factor; not affecting the minima.

The objective function for the expectation vector in the crisp domain is defined as [149],

$$Q_{\vec{x}_\circ}(\vec{x}) = \|\vec{x} - \vec{x}_\circ\|^2. \tag{3.1.4}$$

Thus, the fuzzy expectation is the location of the minimum of the objective function defined as,

$$\Delta_\circ(\vec{x}_\circ) = \sum_{i=1}^{n} p_i \|\vec{x}_i - \vec{x}_\circ\|^2. \tag{3.1.5}$$

Thus,

$$\frac{\partial \Delta_\circ(\vec{x})}{\partial x_i} = 2 \sum_{i=1}^{n} p_i (\vec{x} - \vec{x}_i) = 0 \rightarrow x_i = \frac{1}{\sum_{j=1}^{N} p_j} \sum_{j=1}^{N} p_j \vec{x}_{ji}. \tag{3.1.6}$$

Substituting this value in the objective function defined in (3.1.5) we have,

$$\Delta_\circ \left(\frac{1}{\sum_{j=1}^{N} p_j} \sum_{j=1}^{N} p_j \vec{x}_j \right) = \sum_{i=1}^{n} p_i \left\| \vec{x}_i - \frac{1}{\sum_{j=1}^{N} p_j} \sum_{j=1}^{N} p_j \vec{x}_j \right\|^2, \tag{3.1.7}$$

$$\Delta_\circ \left(\frac{\sum_{j=1}^{N} p_j \vec{x}_j}{\sum_{j=1}^{N} p_j} \right) = \sum_{i=1}^{n} p_i \|\vec{x}_i\|^2 - \frac{\left\| \sum_{j=1}^{N} p_j \vec{x}_j \right\|^2}{\sum_{j=1}^{N} p_i} \leq \sum_{i=1}^{n} p_i \|\vec{x}_i\|^2 = \Delta_\circ(\vec{0}). \tag{3.1.8}$$

Thus, the objective function gives a value smaller than a marginal value in this extremum point, while the function has no more extremum (neither locally nor globally). Hence, this extremum is actually the global minimum. This results in the definition of the fuzzy expectation vector in (3.1.6).

The objective function for the crisp first principal direction is the direction of the maximum deviation or equivalently the direction of minimum one–dimensional reconstruction error, defined as [149],

$$Q_{\vec{x}_\circ}(\vec{x}) = \|\vec{x} - \vec{\eta} - \vec{x}_\circ^T(\vec{x} - \vec{\eta})\vec{x}_\circ\|^2. \tag{3.1.9}$$

Thus, in the fuzzy domain, we should minimize the objective function defined as,

$$\Delta_I(\vec{x}_\circ) = \sum_{i=1}^{n} p_i\|\vec{x}_i - \vec{\eta} - \vec{x}_\circ^T(\vec{x}_i - \vec{\eta})\vec{x}_\circ\|^2. \tag{3.1.10}$$

Note that $\vec{x}_\circ$ satisfies the size constraint $\|\vec{x}_\circ\|^2 = 1$. Then we have,

$$\Delta_I(\vec{x}_\circ) = \sum_{i=1}^{n} p_i(\vec{x}_i - \vec{\eta} - \vec{x}_\circ^T(\vec{x}_i - \vec{\eta})\vec{x}_\circ)^T(\vec{x}_i - \vec{\eta} - \vec{x}_\circ^T(\vec{x}_i - \vec{\eta})\vec{x}_\circ), \tag{3.1.11}$$

$$\begin{aligned}\Delta_I(\vec{x}_\circ) = \sum_{i=1}^{n} p_i\Big[&(\vec{x}_i - \vec{\eta})^T(\vec{x}_i - \vec{\eta}) + \vec{x}_\circ^T(\vec{x}_i - \vec{\eta})^T\vec{x}_\circ\vec{x}_\circ^T(\vec{x}_i - \vec{\eta})\vec{x}_\circ - \\ &\vec{x}_\circ^T(\vec{x}_i - \vec{\eta})(\vec{x}_i - \vec{\eta})^T\vec{x}_\circ - \vec{x}_\circ^T(\vec{x}_i - \vec{\eta})\vec{x}_\circ^T(\vec{x}_i - \vec{\eta}),\Big]\end{aligned} \tag{3.1.12}$$

$$\Delta_I(\vec{x}_\circ) = \sum_{i=1}^{n} p_i\|\vec{x}_i - \vec{\eta}\|^2 - \vec{x}_\circ^T\tilde{C}\vec{x}_\circ, \tag{3.1.13}$$

$$\tilde{C} = \frac{1}{\sum_{i=1}^{N} p_i}\sum_{i=1}^{N} p_i(\vec{x}_i - \vec{\eta})(\vec{x}_i - \vec{\eta})^T. \tag{3.1.14}$$

As the first term in $\Delta_I(\vec{x}_\circ)$ is not a function of $\vec{x}_\circ$, thus, $\Delta_I(\vec{x}_\circ)$ is minimized when the second term is maximized. We use the method of Lagrange multipliers to embed the size constraint in the objective function. Thus, the function to be maximized is,

$$\bar{\Delta}_I(\vec{x}_\circ) = \vec{x}_\circ^T \tilde{C} \vec{x}_\circ - \lambda(\|\vec{x}_\circ\|^2 - 1). \tag{3.1.15}$$

Hence,

$$\frac{\partial \bar{\Delta}_I(\vec{x})}{\partial \vec{x}} = 2\tilde{C}\vec{x} - 2\lambda\vec{x} = 0 \rightarrow \tilde{C}\vec{x} = \lambda\vec{x}. \tag{3.1.16}$$

Thus, $\vec{x}$ is the eigenvector of $\tilde{C}$ corresponding to the largest eigenvalue. Again note that for this value of $\vec{x}$ we have,

$$\Delta_I(\vec{x}) = \sum_{i=1}^{n} p_i \|\vec{x}_i - \vec{\eta}\|^2 - \vec{x}^T \tilde{C} \vec{x} = \sum_{i=1}^{n} p_i \|\vec{x}_i - \vec{\eta}\|^2 - \lambda \vec{x}^T \vec{x}, \tag{3.1.17}$$

$$\Delta_I(\vec{x}) = \sum_{i=1}^{n} p_i \|\vec{x}_i - \vec{\eta}\|^2 - \lambda < \sum_{i=1}^{n} p_i \|\vec{x}_i - \vec{\eta}\|^2 = \Delta_I(\vec{0}). \tag{3.1.18}$$

Hence, the value of the objective function in this point is smaller than it's value in $\vec{0}$ (a marginal point), hence, it is the global minima.

It can be easily proved (using the same method) that the other fuzzy principal directions correspond to the other eigenvectors of $\tilde{C}$, sorted by the corresponding eigenvalues in a descending fashion. This is the same as the classic PCA when replacing the covariance matrix with the fuzzy covariance matrix.

CIP includes the function *fpca* which is developed for performing the FPCA transform in 3–D spaces (See Figure 3.1). The function *fpca* inputs a color image

fpca

```
function [E,V,D]=FPCA(I,M)
%FPCA computes fuzzy principle components of the image I according to
%the membership mask M.
%
%Usage:[E,V,D]=FPCA(I,M)
%Inputs: I: Input image in RGB, must be of type Uint8 .
%          If using imread, be sure to load a 24bit bmp file
%          Never use indexed images for I.
%       M: Membership mask.
%Outputs: E: Expectation value for the three channels. 3*1 of type double
%           in range [0,255]
%         V :Principle components. V(:,i) gives the i-th principle
%         component
%         D:Eigen values of the convariance matrix
%
%FPCA stands for Fuzzy Principle Component Analysis
%Revision 1.0 (c) Arash Abadpour Fall 2004 (Oct. 5, 2004)
```

Figure 3.1: Implementation of the *fpca* function.

along with a fuzzy map. Then it outputs the fuzzy expectation vector along with the principal matrix and the eigenvalue matrix.

3.1.2 Weighted Powered Sum Minimization

Consider the optimization problem defined as minimizing the function,

$$\Delta(x_1, \cdots, x_n) = \sum_{i=1}^{n} w_i x_i^m, \tag{3.1.19}$$

assuming,

$$\sum_{i=1}^{n} x_i = 1, \tag{3.1.20}$$

for known values of $w_1, \cdots, w_n$. We propose to substitute,

$$x_p = 1 - \sum_{i=1, i\neq p}^{n} x_i, \tag{3.1.21}$$

in $\Delta(x_1, \cdots, x_n)$ to have,

$$\Delta = \sum_{i=1, i\neq p}^{n} w_i x_i^m + w_p\Big(1 - \sum_{i=1, i\neq p}^{n} x_i\Big)^m, \tag{3.1.22}$$

$$\frac{\partial \Delta}{\partial x_j}_{j\neq p} = m w_j x_j^{m-1} - m w_p\Big(1 - \sum_{i=1, i\neq p}^{n} x_i\Big)^{m-1} = 0. \tag{3.1.23}$$

Thus,

$$w_j x_j^{m-1} = w_p x_p^{m-1} \rightarrow x_j = \frac{w_p^{\frac{1}{m-1}}}{w_j^{\frac{1}{m-1}}} x_p = \frac{w_j^{-\frac{1}{m-1}}}{w_p^{-\frac{1}{m-1}}} x_p, j \neq p. \tag{3.1.24}$$

Hence,

$$\sum_{j=1, j\neq p}^{n} x_j + x_p = \left[\sum_{j=1, j\neq p}^{n} \frac{w_j^{-\frac{1}{m-1}}}{w_p^{-\frac{1}{m-1}}} + 1\right] x_p, \tag{3.1.25}$$

and[1],

$$1 = \left[\sum_{j=1}^{n} \frac{w_j^{-\frac{1}{m-1}}}{w_p^{-\frac{1}{m-1}}}\right] x_p \rightarrow x_p = \frac{w_p^{-\frac{1}{m-1}}}{\sum_{j=1}^{n} w_j^{-\frac{1}{m-1}}}. \tag{3.1.30}$$

[1]A simpler solution comes from applying Lagrange multipliers. Define,

$$\tilde{\Delta} = \sum_{i=1}^{n} w_i x_i^m - \lambda(\sum_{i=1}^{n} x_i - 1). \tag{3.1.26}$$

Thus,

$$\frac{\partial \tilde{\Delta}}{\partial x_i} = m w_i x_i^{m-1} - \lambda = 0 \rightarrow x_i = \lambda m^{-\frac{1}{m-1}} w_i^{-\frac{1}{m-1}}, \tag{3.1.27}$$

Note that with the above definition we have,

$$\sum_{p=1}^{n} x_p = \frac{\sum_{p=1}^{n} w_p^{-\frac{1}{m-1}}}{\sum_{j=1}^{n} w_j^{-\frac{1}{m-1}}} = 1. \tag{3.1.31}$$

Thus, the normality constraint is satisfied. Substituting x_p in $\Delta(x_1, \cdots, x_n)$ we have,

$$\Delta = \sum_{p=1}^{n} w_p \frac{w_p^{-\frac{m}{m-1}}}{\left(\sum_{j=1}^{n} w_j^{-\frac{1}{m-1}}\right)^m} = \left(\sum_{j=1}^{n} w_j^{-\frac{1}{m-1}}\right)^{-(m-1)}, \tag{3.1.32}$$

which is the mean of order $-\frac{1}{m-1}$ of $w_1, \cdots, w_n$. Now,

$$\forall i, w_i > 0 \rightarrow w_i^{-\frac{1}{m-1}} > 0 \rightarrow \sum_{j=1}^{n} w_j^{-\frac{1}{m-1}} > w_i^{-\frac{1}{m-1}}, \tag{3.1.33}$$

$$\forall i, \left(\sum_{j=1}^{n} w_j^{-\frac{1}{m-1}}\right)^{-(m-1)} < w_i. \tag{3.1.34}$$

Setting all x_j equal to zero except for $x_i = 1$ satisfies the normality constraint and gives $\Delta = w_i$. Thus, the relation in (3.1.30) is the global minima of the objective function. We use this concept in the next section to propose the general fuzzy clustering algorithm.

$$1 = \sum_{i=1}^{n} x_i = \lambda m^{-\frac{1}{m-1}} \sum_{i=1}^{n} w_i^{-\frac{1}{m-1}} \rightarrow \lambda m^{-\frac{1}{m-1}} = \frac{1}{\sum_{i=1}^{n} w_i^{-\frac{1}{m-1}}}. \tag{3.1.28}$$

Hence,

$$x_p = \frac{w_p^{-\frac{1}{m-1}}}{\sum_{i=1}^{n} w_i^{-\frac{1}{m-1}}}, \tag{3.1.29}$$

which is the same as (3.1.30).

3.1.3 General Fuzzy Clustering Algorithm

Consider the general fuzzy clustering problem stated as minimizing the objective function defined as,

$$J(X, \Phi) = \sum_{i=1}^{n} \sum_{j=1}^{c} p_{ij}^{m} D_{ij}. \tag{3.1.35}$$

This problem describes the best choice for clustering the members of the set $X = \{\vec{x}_1, \cdots, \vec{x}_n\}$ into c clusters described as $\Phi = \{\phi_1, \cdots, \phi_c\}$. Here, p_{ij} is the fuzzy membership of $\vec{x}_i$ to the j–th cluster and D_{ij} is the distance between this point and the j–th cluster. Assume that $D_{ij} = \Psi(\vec{x}_i, \phi_j)$ is the appropriate distance function for the vector geometry under investigation. As an example, as discussed in Section 3.3, the LPRE is the proper distance for color vectors. Consequently, under the Ψ distance function, we have,

$$p_{ij} \propto D_{ij}, p_{ij} \propto D_{pj}^{-1}, 1 \leq p \leq n, p \neq i. \tag{3.1.36}$$

Here, ϕ_j is the defining parameters of the j–th cluster according to the general cluster model. Again, as an example in the LPRE methodology, $\phi_j = [\vec{\eta}_j, \vec{v}_j]$.

As a special case, the FCM [168] models each cluster with a single vector ($\phi_j = [\vec{\eta}_j]$), and defines the Ψ function as the squared Euclidean distance between the given vector and the cluster center. The key point of the proposed general fuzzy clustering algorithm is the deep relation between the cluster model and the cluster tuning function. For a set of given points, the expectation vector minimizes the sum of the

squared Euclidean distances.

Assume that the function Υ tunes the cluster model ϕ to best fit the fuzzy vectors in $\tilde{X}$. This means that for the fuzzy set of vectors $\tilde{X} = \{(\vec{x}_i; p_i)|i = 1, \cdots, n\}$,

$$\Upsilon(\tilde{X}) = \arg_{\phi} \min \left\{ \sum_{i=1}^{n} p_i \Psi(\vec{x}_i, \phi) \right\}. \tag{3.1.37}$$

Therefore, the function $\Upsilon(\tilde{X}) \equiv \tilde{E}\{\tilde{X}\}$ is the solution to $\Psi(\vec{x}, \phi) = \|\vec{x} - \vec{\eta}\|^2$. Here, $\tilde{E}\{\tilde{X}\}$ stands for the fuzzy expectation of a fuzzy set. Also, the FPCA introduced in Section 3.1.1 gives the $\Upsilon(\tilde{X})$ function corresponding to the Ψ function defined as the LPRE.

Back to the main problem of minimizing $J(X, \Phi)$, assume that we have the dual functions, $\Psi(\cdot)$ and $\Upsilon(\cdot)$. Here, we propose an algorithm that converges to a minimal point of $J(X, \Phi)$, if at least one exists. Consider rewriting,

$$J(X, \Phi) = \sum_{i=1}^{n} \Delta_i, \Delta_i = \sum_{j=1}^{c} p_{ij}^m D_{ij}. \tag{3.1.38}$$

Also, assume that we have fixed D_{ij} and we are trying to decline $J(X, \Phi)$ by working on p_{ij}. As the only restriction on p_{ij} is the normality condition,

$$\forall i, \sum_{j=1}^{c} p_{ij} = 1, \tag{3.1.39}$$

there is no connection between p_{ij} and $p_{i'j}$, for $i \neq i'$. Thus, declining Δ_is independently results in declination of $J(X, \Phi)$. Considering the normality constraints, minimizing Δ_i is a weighted powered sum minimization (discussed in Section 3.1.2).

Hence, we get,

$$p_{ij} = \frac{D_{ij}^{-\frac{1}{m-1}}}{\sum_{k=1}^{c} D_{ik}^{-\frac{1}{m-1}}}. \tag{3.1.40}$$

During this stage, $J(X, \Phi)$ is declined, except for the case that p_{ij}s satisfy the above equation at first. We will come back to this situation later.

Now, consider rewriting,

$$J(X, \Phi) = \sum_{j=1}^{c} \Theta_j, \Theta_j = \sum_{i=1}^{n} p_{ij}^m D_{ij}. \tag{3.1.41}$$

Assume that we have fixed p_{ij} and we are trying to decline $J(X, \Phi)$ by working on D_{ij}. Declining Θ_j means tuning the j–th cluster to minimize the overall distances in a fuzzy scheme. Note that like above, it can be solved independently for different clusters, resulting in overall decline of $J(X, \Phi)$. The solution in this state is obtained by using the Υ function as,

$$\phi_j = \Upsilon\left(\left\{(\vec{x}_i, p_{ij}^m)|i = 1, \cdots, n\right\}\right). \tag{3.1.42}$$

Again, $J(X, \Phi)$ can never increase in this state, and it's constancy shows that all the clusters have been in the best place, according to the special distance function Ψ.

The halting test of the algorithm is the stationarity of the fuzzy membership maps. Assume that algorithm has passed through l iterations. Also, assume that F_{ijk} is the membership of $\vec{x}_k$ to the j–th cluster in the i–th iteration. Now the difference between the result of the i–th iteration and the j–th iteration is defined as,

$$\delta_{i,j} = \sqrt{\frac{1}{WHc} \sum_{k=1}^{n} \sum_{l=1}^{c} (F_{ilk} - F_{jlk})^2}. \tag{3.1.43}$$

Also, the repeatedness of the result of the i–th iteration is defined as,

$$\delta_i = \min\left\{\delta_{i,1}, \cdots, \delta_{i,i-1}\right\}. \tag{3.1.44}$$

Now, the condition for ending the iterations is $\delta_i < \delta$ where δ is a predefined threshold. Using $\delta = 0.05$ is a reasonable choice which means that the algorithm will stop iterating when fuzzy membership values repeat with less than 5% variation.

Note that $\{\phi_1, \cdots, \phi_c\}$ are randomly selected at the beginning of the algorithm. During two consecutive stages, $J(X, \Phi)$ never rises and the case of it being constant in two consecutive stages results in remaining constant for all coming stages. Thus, using this method, $J(X, \Phi)$ is going towards a minimum point, and never gets oscillated. Note that along with the FCM, other methods like GK [161], FEC [236], and FCV [173], are also special cases of the proposed general fuzzy clustering method. Figure 3.2 shows the pseudo–code of the proposed algorithm.

3.1.4 Basis Vector Polarization

Consider the space R^n and a set of n basis vectors $\vec{v}_i, i = 1, \cdots, n$. Storing this set of vectors needs n^2 units of memory (when neglecting the redundancy). Having in mind that a set of basis vectors are an ortho–normal set, the actual needed memory can be reduced. In fact, a set of basis vectors of R^n is a member of R^{n^2}, with n constraints of normality ($\|\vec{v}_i\| = 1, i = 1, \cdots, n$) and $\frac{n(n-1)}{2}$ constraints of orthogonality ($\vec{v}_i \perp \vec{v}_j, i, j = 1, \cdots, n, i \neq j$). Thus, the above–mentioned set of the basis vectors is

- ***Aim***: *Clustering data according to the given model.*
- ***Inputs***:
  - *Appropriate distance function* Ψ *and its dual* Υ.
  - *Set of realizations* $X = \{\vec{x}_1, \cdots, \vec{x}_n\}$.
  - *Number of clusters* c.
  - *Fuzzyness* m.
  - *Halting threshold* δ.
- ***Output***:
  - *Fuzzy membership values* $p_{ij}, 1 \leq i \leq n, 1 \leq j \leq c$.
  - c *cluster descriptors* $\phi_1, \cdots, \phi_c$.
- ***Method***:

  1 $l = 0$.

  2 *Randomize* $\phi_1, \cdots, \phi_c$.

  3 $l = l + 1$.

  4 *Compute distances as* $D_{ij} = \Psi(\vec{x}_i, \phi_j), 1 \leq i \leq n, 1 \leq j \leq c$.

  5 *Compute fuzzy membership values as* $p_{ij} = D_{ij}^{-1/m-1} / \sum_{k=1}^{c} D_{ik}^{-1/m-1}, 1 \leq i \leq n, 1 \leq j \leq c$.

  6 *Store the fuzzification scheme as* $F_{lji} = p_{ij}, 1 \leq i \leq n, 1 \leq j \leq c$.

  7 *Renew clusters as* $\phi_j = \Upsilon\Big\{(\vec{x}_1; p_{1j}^m), \cdots, (\vec{x}_n; p_{nj}^m)\Big\}$.

  8 *if* $l > 1$ *compute repeatedness as* $\delta_l = \min_{1 \leq p \leq l-1} \left\{ \sqrt{\frac{1}{WHc} \sum_{k=1}^{n} \sum_{j=1}^{c} (F_{pjk} - F_{ljk})^2} \right\}$.

  9 *if* $l = 1$ *then goto* 3.

  10 *if* $l > 1$ *and* $\delta_l > \delta$ *then goto* 3 *else return* $p_{ij}, 1 \leq i \leq n, 1 \leq j \leq c$ *and* $\phi_i, 1 \leq i \leq c$.

Figure 3.2: Pseudo–code of the proposed general fuzzy clustering algorithm.

an unconstrained member of an m–D space, with m equal to:

$$m = n^2 - n - \frac{n(n-1)}{2} = \frac{n(n-1)}{2}. \tag{3.1.45}$$

Hence, storing a set of basis vectors of R^n in $\frac{n(n-1)}{2}$ memory cells contains zero redundancy. To make this representation unique, it is crucial to make the set *right–rotating* (RR). In R^2, RR means:

$$(\vec{v}_1 \times \vec{v}_2) \cdot \vec{j} > 0, \tag{3.1.46}$$

where, $\times$ and $\cdot$ denote the outer and the inner products, respectively. In R^3, RR means:

$$(\vec{v}_1 \times \vec{v}_2) \cdot \vec{v}_3 > 0. \tag{3.1.47}$$

Setting $n = 2$ in (3.1.45) leads to $m = 1$, which means that any set of RR basis vectors in the xy plane can be specified uniquely by a single parameter (the angle). Similarly, the case of $n = 3$ results in $m = 3$, which is used in this thesis. We will prove that the parameters in the 3–D case are angular too. Thus, we call this method of representing a set of basis vectors, the polarization method. Here we propose a method for finding these angles.

Consider the three RR vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$ in R^3. We define the angles θ, ϕ, and ψ as a manipulated version of the well–known set of Euler angles. Using $\vec{v}^p$ as the projection of $\vec{v}$ on plane p (*e.g.*, $\vec{v}_1^{xy}$), the three angles are defined as:

$$\theta = \angle(\vec{v}_1^{xy}, [1, 0]^T), \tag{3.1.48}$$

vecangle
function theta=vecangle(v1,v2) *%Function vecangle computes the angle between two vectors v1 and v2 in* *%radians.* *%Usage: vecangle(v1,v2)* *%Inputs: v1,v2:Input vectors (2*1).* *%Outputs: Theta: The angle between v1 and v2 (-pi..pi):* *% How much should v1 rotate to reach v2* *%Revision 1.21 Arash Abadpour Fall 2004 (Dec. 10, 2004)*

Figure 3.3: Implementation of the *vecangle* function.

$$\phi = \angle((R_\theta^{xy}\vec{v}_1)^{xz}, [1,0]^T), \tag{3.1.49}$$

$$\psi = \angle((R_\phi^{xz}R_\theta^{xy}\vec{v}_2)^{yz}, [1,0]^T), \tag{3.1.50}$$

where, $\angle(\vec{v},\vec{u})$ denotes the angle between two vectors $\vec{v},\vec{u} \in R^2$, computed as:

$$\angle(\vec{v},\vec{u}) = sgn((\vec{v}\times\vec{u})\cdot\vec{j})\cos^{-1}\frac{\vec{v}\cdot\vec{u}}{\|\vec{v}\|\|\vec{u}\|}. \tag{3.1.51}$$

The above equation is developed in the function *vecangle* shown in Figure 3.3. Here, $sgn(x)$ is the signum function, defined as:

$$sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}. \tag{3.1.52}$$

Also, R_α^p is the 3×3 matrix of α radians rotated counter–clock–wise in the p

plane:

$$R_{\theta}^{xy} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.1.53}$$

$$R_{\phi}^{xz} = \begin{pmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{pmatrix}, \tag{3.1.54}$$

$$R_{\psi}^{yz} = \begin{pmatrix} 0 & \cos\psi & -\sin\psi \\ 1 & 0 & 0 \\ 0 & \sin\psi & \cos\psi \end{pmatrix}. \tag{3.1.55}$$

Composing the 3×3 matrix V with $\vec{v_i}$ as its i–th column, we get:

$$R_{\psi}^{yz} R_{\phi}^{xz} R_{\theta}^{xy} V = I. \tag{3.1.56}$$

Having in mind that,

$$(R_{\alpha}^{p})^{-1} = R_{-\alpha}^{p}, \tag{3.1.57}$$

yields to,

$$V = R_{-\theta}^{xy} R_{-\phi}^{xz} R_{-\psi}^{yz}. \tag{3.1.58}$$

While equations (3.1.48), (3.1.49), and (3.1.50) compute the three angles θ, ϕ, and ψ out of the basis vectors (polarization), equation (3.1.58) reproduces the base from θ, ϕ, and ψ angles (depolarization). Using the notations of the compression theory, this method gives a compression ratio of $3:1$ in R^3 (and $\frac{2n}{n-1}:1$ in R^n).

compbv
function [theta,phi,psi]=compbv(V) *%function compress basis vector compresses the base vector present in V as* *%columns to three angles.* *%Usage: [theta,phi,psi]=compbv(V)* *%Inputs: V:The 3*3 matrix contining the base vectors as its columns.* *%Outputs: theta,phi,psi: The Euler angles.* *%Revision 1.0, Arash Abadpour, Spring 2004.*

Figure 3.4: Implementation of the *compbv* function.

CIP contains two functions *compbv* and *decompbv* for polarizing and depolarizing in R^3, respectively (see Figures 3.4 and 3.5). It also contains the function *makerr* for giving the right–rotating representation of a given matrix (see Figure 3.6).

3.1.5 Block–wise Interpolation

Consider a partition of the $N_W \times N_H$ as a set of rectangular regions $\{r_i | i = 1, \cdots, n\}$, with corresponding (given) values of $\{\lambda_i | i = 1, \cdots, n\}$, satisfying,

$$\forall i, \forall \vec{c} \in r_i, f(\vec{c}) \simeq \lambda_i, \tag{3.1.59}$$

for an arbitrary function $f : R^2 \rightarrow R$. The problem is to find $\tilde{f}$ as a good approximation of f. We address this problem as block–wise interpolation of the set $\{(r_i; \lambda_i) | i = 1, \cdots, n\}$. Note that in the case that the partition is a conventional rectangular grid, the problem reduces to an ordinary 2–D interpolation task. Here, we use a reformulated version of the well–known low–pass Butterworth filter, as the

decompbv

```
function V=decompbv(theta,phi,psi)
%Function decompress base vectors computes the matrix V containing the base
%vectors as its columns from the three angles theta, phi, and psi.
%Usage:      V=decompbv(theta,phi,psi);
%Inputs:     theta,phi,psi: The Euler angles.
%Outputs:    V:The 3*3 matrix contining the base vectors as its columns.
%Revision 1.001, Arash Abadpour, Winter 2004 (March 3, 2005).
```

Figure 3.5: Implementation of the *decompbv* function.

makerr

```
function U=makeRR(V)
%Function make right rotating, changes the set of basis vectors available in
%V as columns to a set of right rotating base vectors.
%Usage: U=makeRR(V);
%Inputs:     V:The 3*3 matrix having the base vectors as its columns.
%Outputs:    U:The resulting 3*3 matrix having the base vectors as its columns.
%Revision 1.0, Arash Abadpour, Spring 2004.
```

Figure 3.6: Implementation of the *makerr* function.

interpolation kernel,

$$B_{\tau,N}(x) = \left(1 + \left(\frac{x}{\tau}\right)^{2N}\right)^{-\frac{1}{2}}, \tag{3.1.60}$$

where N and τ are defined as:

$$N = rnd\left(\log_{\frac{a}{b}}\left(\frac{\beta\sqrt{1-\alpha^2}}{\alpha\sqrt{1-\beta^2}}\right)\right), \tag{3.1.61}$$

$$\tau = a\sqrt[2N]{\frac{\alpha^2}{1-\alpha^2}}. \tag{3.1.62}$$

The function $B_{\tau,N}(\cdot)$ satisfies the conditions of $B_{\tau,N}(a) = \alpha$ and $B_{\tau,N}(b) = \beta$. The 2–D version of this function is defined as:

$$B_{\tau,N,w,h}(x,y) = B_{w\tau,N}(x)B_{h\tau,N}(y), \tag{3.1.63}$$

where w and h control the spread of the function in the x and y directions, respectively. Figure 3.7 shows the typical shape of the function $B^{w,h}_{\tau,N}(x,y)$ with $a = 0.9$, $\alpha = 0.7$, $b = 1$, $\beta = 0.5$, and $w = h = 16$.

Assuming that the region r_i is centered at (x_i, y_i), while its height and width are w_i and h_i, respectively, we propose the function $\tilde{f}$ to be defined as:

$$\tilde{f}(x,y) = \frac{\sum_{i=1}^{N} \lambda_i B_{\tau,N,\frac{w_i}{2},\frac{h_i}{2}}(x - x_i, y - y_i)}{\sum_{i=1}^{N} B_{\tau,N,\frac{w_i}{2},\frac{h_i}{2}}(x - x_i, y - y_i)}. \tag{3.1.64}$$

Now, $\tilde{f}(x,y)$ is a smooth version of the initial step case function:

$$\forall [x,y]^T \in r_i : f_\circ(x,y) = \lambda_i. \tag{3.1.65}$$

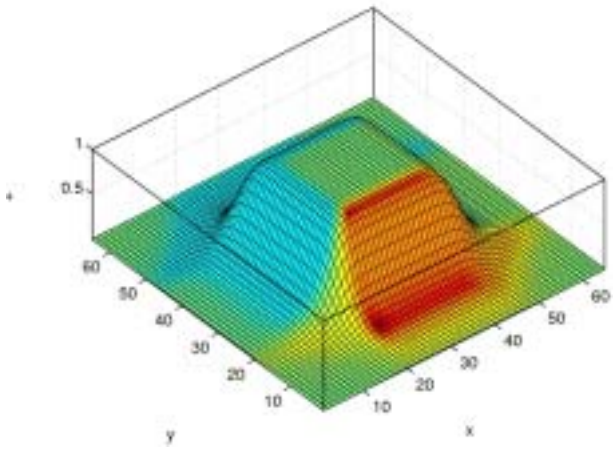


Figure 3.7: Typical shape of the proposed interpolation kernel, $B^{w,h}_{\tau,N}(x,y)$.

Also, by setting proper values of the parameters a, b, α, and β, the function $\tilde{f}(x,y)$ will satisfy (3.1.59). The proper set of parameters must force $B^{\frac{w_i}{2},\frac{h_i}{2}}_{\tau,N}(x-x_i, y-y_i)$ to become nearly one in entire r_i (except for the borders neighborhoods) and also to prevent r_i to intrude the interior of r_j, for $i \neq j$. Selecting a value near 1 (but smaller) for a and α, limits the declination of the ceil of the function, while setting $b = 1$ and a not too big value for β controls the effects of neighbor regions on each other. Note that setting $a = 1^-$, $\alpha = 1$, $b = 1^+$, and $\beta = 0$, is the marginal choice leading to no smoothing (the same as (3.1.65)).

As the generalization of the block–wise interpolation problem, assume the set of regions $\{(r_i;\lambda_{ij})|i = 1,\cdots,n, j = 1,\cdots,m\}$, satisfying,

$$\lambda_{ij} = \arg_\lambda \left(\forall \vec{c} \in r_i, f_j(\vec{c}) \simeq \lambda\right), \tag{3.1.66}$$

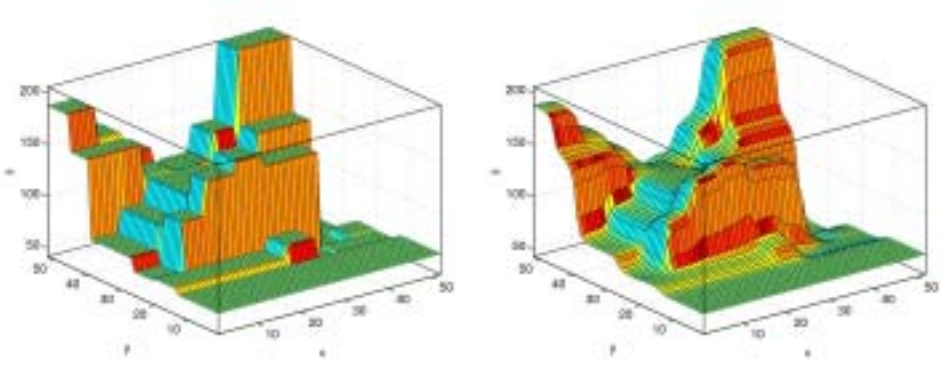
for a set of arbitrary functions $f_i : R^2 \rightarrow R, i = 1, , \cdots , m$. In a similar manner with (3.1.64), we propose:

$$\tilde{f}_j(x, y) = \frac{\sum_{i=1}^{n} \lambda_{ij} B_{\tau,N,\frac{w_i}{2},\frac{h_i}{2}}(x - x_i, y - y_i)}{\sum_{i=1}^{n} B_{\tau,N,\frac{w_i}{2},\frac{h_i}{2}}(x - x_i, y - y_i)}. \tag{3.1.67}$$

Here, because the set of the base regions for all $\tilde{f}_j$ are the same, the total performance is increased by computing $B_{\tau,N}^{\frac{w_i}{2},\frac{h_i}{2}}(x - x_i, y - y_i)$ for each value of i, just once. Then, the problem reduces to m times computation of a weighted average.

When working in the polar coordinates, because of the 2π discontinuity, ordinary algebraic operations on the variables lead to spurious results (for example $\frac{0+2\pi}{2} = \pi$, while the average of 0 radians and 2π radians equals $0 \equiv 2\pi$ radians). To overcome this problem, we propose a new method: for the given problem, $\{(r_i; \theta_i) | i = 1, \cdots , n\}$, solve the problem $\{(r_i; \cos\theta_i, \sin\theta_i) | i = 1, \cdots , n\}$ and then find θ_i using ordinary trigonometric methods. Note that interpolating both $\sin\theta_i$ and $\cos\theta_i$ is performed to avoid ambiguity in the polar plane.

Figure 3.8–a shows a sample problem set given to the proposed block–wise Interpolation. Figure 3.8–b illustrates the corresponding $\tilde{f}$. In this sample the resulting SNR is more than $22dB$. Function *interpr* is using the above scheme for interpolating the expectation vector and the polarized PCA matrix over the entire image (see Section 3.9).



(a) (b)

Figure 3.8: Proposed block–wise interpolation, (a) sample problem and (b) the solution with $SNR > 22dB$.

interpr

```
function [E1,E2,E3,THETA,PHI,PSI]=interpr(r,w,h)
%Function interpolate regions interpolates the region information matrix
%and gives the solution for finding the eigenimages.
%Usage:      [E1,E2,E3,THETA,PHI,PSI]=interpr(r,w,h)
%Inputs:     r:Region information.
%           w,h:Original image size.
%Outputs:    E1,E2,E3:Emap.
%           THETA,PHI,PSI:Rmap.
%Revision 1.0 Arash Abadpour Spring 2004 (1383/2/27)
```

Figure 3.9: Implementation of the *interpr* function.

3.1.6 Butterworth membership function

A distance function give lower values to the color vectors similar to those that exist in r. Thus, a function is needed to map $[-1, 1]$ into $[1, 1-\varepsilon]$ and $[-\infty, -1] \cup [1, \infty]$ into $[1-\varepsilon, 0]$ to give conventional fuzzy membership values. we propose to use a manipulated form of the well–known low–pass Butterworth filter for the sake of tunability and simplicity, as:

$$B_{\alpha,\beta}(x) = \left(1 + \left(\frac{x}{\tau_{\alpha,\beta}}\right)^{2N_{\alpha,\beta}}\right)^{-\frac{1}{2}}, \tag{3.1.68}$$

where, $N_{\alpha,\beta}$ and $\tau_{\alpha,\beta}$ are defined as:

$$N_{\alpha,\beta} = rnd\left(\log_2\left(\frac{\alpha\sqrt{1-\beta^2}}{\beta\sqrt{1-\alpha^2}}\right)\right), \tag{3.1.69}$$

$$\tau_{\alpha,\beta} = \alpha^{\frac{1}{N_{\alpha,\beta}}}\left(1-\alpha^2\right)^{-\frac{1}{2N_{\alpha,\beta}}}, \tag{3.1.70}$$

where $rnd(x)$ denotes the nearest integer value to x. The $B_{\alpha,\beta}(x)$ is designed such that it satisfies $B_{\alpha,\beta}(1) = \alpha$ and $B_{\alpha,\beta}(2) = \beta$. Selecting a large member of $]0, 1[$ as the α value and an small member of $]0, \alpha[$ as the β value, leads to a desired fuzzyfication. Note that the above definition of membership functions is in contrast with the general selection of the Gaussian functions. Figure 3.10 shows the typical shape of the membership function with the two parameters α and β highlighted on it. Figure 3.11 shows the log–magnitude of the membership function ($B_{\alpha,\beta}$) for different values of α and β. While larger values of β increases the domain of the typical

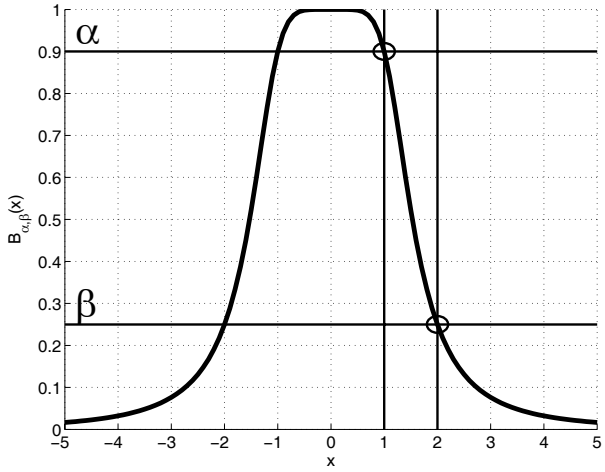


Figure 3.10: Typical shape of the proposed membership function and α and β parameters.

members (those with large likelihood values) the function has an interior of almost–one values for the interior points of the cluster controlled by α. This situation does not happen in the Gaussian function, in which one must increase the domain of better members (by increasing σ) leading to an overall increase of membership values. Also, the sharp transition between the points $(1, \alpha)$ and $(2, \beta)$, which enables the crisp classification of points in the proposed membership functions, does not exist in the Gaussian function.

CIP includes two function *bw* and *butterworth* for computing τ and N from given values of a and b, and for applying the membership function on data, respectively. The function *bw* which is shown in Figure 3.12 gives a and b and computes N and T, satisfying $B_{T,N}(1) = a$ and $B_{T,N}(2) = b$. The function butterworth computes

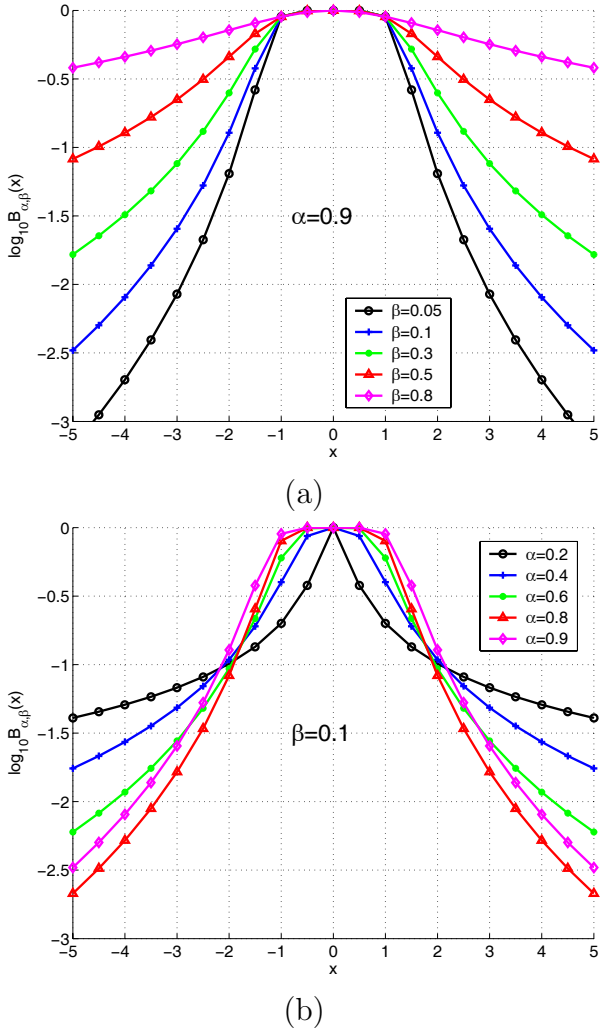


Figure 3.11: Log–magnitude of the membership function ($B_{\alpha,\beta}$). (a) $\alpha = 0.9$ with different values of β. (b) $\beta = 0.1$ with different values of α.

bw

```
function [N,T]=BW(a,b)
%BW computes the parameters of a butterworth membership function that satisfy
%f(1)=a and f(2)=b.
%Usage:[N,T]=BW(a,b)
%Inputs:     a:f(1)
%            b:f(2)
%Outputs:    N:Order
%            T:Constant
%Note : f(x)=(1+(x/T)^(2*N))^-0.5
%Revision 1.0 Arash Abadpour Winter 2004
```

Figure 3.12: Implementation of the *bw* function.

the butterworth function for an arbitrary array, given that T and N are known (see Figure 3.13).

3.1.7 Eigenimages

Consider the PCA matrix, V_r, and the expectation vector, $\vec{\eta}_r$, corresponding to the homogenous cluster r. Then, for the color vector $\vec{c}$ belonging to r we get the PCA coordinates as,

$$\vec{c}_{pc} = V_r^{-1}(\vec{c} - \vec{\eta}_r). \tag{3.1.71}$$

Assume that we can somehow find the color cluster $r_{\vec{c}}$ for each color vector $\vec{c}$, where $r_{\vec{c}}$ describes the color content of $\vec{c}$, in the sense that,

$$\vec{c}_{pc} = V_{r_{\vec{c}}}^{-1}(\vec{c} - \vec{\eta}_{r_{\vec{c}}}), \tag{3.1.72}$$

butterworth

```
function Y=ButterWorth(X,T,N)
%Function butterworth computes the butter worth filter of order N and
%constant T applied on X.
%Usage:     Y=ButterWorth(X,T,N)
%Inputs:    X:Input data
%           T:Constant
%           N:Order
%Output:    Y:Result
%See also BW.
%Revision 1.0 Arash Abadpour Winter 2004
```

Figure 3.13: Implementation of the *butterworth* function.

satisfies:

$$\sigma_{pc_1} \gg \sigma_{pc_2} \gg \sigma_{pc_3}, \tag{3.1.73}$$

where $\vec{c}_{pc} = [c_{pc_1}, c_{pc_2}, c_{pc_3}]^T$. We call the pc_1, pc_2, and pc_3 images as the eigenimages. The original image can be perfectly reconstructed using these three channels, except for the numerical errors,

$$\vec{c} \simeq \vec{c}_3 = V_{r_{\vec{c}}} \vec{c}_{pc} + \vec{\eta}_{r_{\vec{c}}}. \tag{3.1.74}$$

It is shown in Section 3.3 that for homogenous swatches, neglecting pc_3 (or even both pc_2 and pc_3), gives good approximations of the original image. Here, we generalize the approach. Note that the perfect reconstruction scheme of (3.1.74) does not rely

on (3.1.72), while the partial reconstructions defined as:

$$\vec{c}_2 = V_{r_{\vec{c}}} \begin{pmatrix} c_{pc_1} \\ c_{pc_2} \\ 0 \end{pmatrix} + \vec{\eta}_{r_{\vec{c}}}, \tag{3.1.75}$$

and,

$$\vec{c}_1 = V_{r_{\vec{c}}} \begin{pmatrix} c_{pc_1} \\ 0 \\ 0 \end{pmatrix} + \vec{\eta}_{r_{\vec{c}}}, \tag{3.1.76}$$

do rely on (3.1.72).

Although (3.1.76) gives a 1–D representation of a given color image, if the computation of $V_{r_{\vec{c}}}$ and $\vec{\eta}_{r_{\vec{c}}}$ gets expensive the scheme, although being theoretically promising, is actually not practical. Thus, we seek for a method for describing $V_{r_{\vec{c}}}$ and $\vec{\eta}_{r_{\vec{c}}}$ in a simple way.

From linear algebra we know that for orthonormal matrices V_r the transformation of equation (3.1.72) satisfies[2]

$$\sigma^2_{pc_1} + \sigma^2_{pc_2} + \sigma^2_{pc_3} = \sigma^2_r + \sigma^2_g + \sigma^2_b. \tag{3.1.80}$$

[2]First, note that,

$$\sum_{i=1}^{3} c^2_{pc_i} = \vec{c}^T_{pc}\vec{c}_{pc} = (\vec{c} - \vec{\eta})^T V V^{-1}(\vec{c} - \vec{\eta}) = \sum_{i=1}^{3}(c_i - \eta_i)^2. \tag{3.1.77}$$

Also,

$$E\{\vec{c}_{pc}\} = V^{-1}E\{\vec{c} - \vec{\eta}\} = \vec{0}. \tag{3.1.78}$$

Hence,

$$\sum_{i=1}^{3} \sigma^2_{pc_i} = E\Big\{\sum_{i=1}^{3} c^2_{pc_i}\Big\} = E\Big\{\sum_{i=1}^{3}(c_i - \eta_i)^2\Big\} = \sum_{i=1}^{3}\sigma^2_{c_i}. \tag{3.1.79}$$

Thus,

$$\kappa_x = \frac{\sigma_x^2}{\sigma_{pc_1}^2 + \sigma_{pc_2}^2 + \sigma_{pc_3}^2}, \tag{3.1.81}$$

where x is one of the r, g, b, pc_1, pc_2, and pc_3 shows the amount of information available in the x channel. Hence, we have,

$$\kappa_1 + \kappa_2 + \kappa_3 = \kappa_r + \kappa_g + \kappa_b = 1. \tag{3.1.82}$$

The larger the κ_x, the more important we expect the corresponding channel to be. We expect $\kappa_{pc_1} \gg \kappa_{pc_2} \gg \kappa_{pc_3}$ to make it reasonable to eliminate pc_2 and pc_3 while saving the image quality.

3.1.8 Uniform Quantized Random Variable

Assume that the random variable r is given which is uniform in the range $(0, 1]$. We compute the new random variable s as,

$$s = A\left(2\frac{\lceil 2^q r \rceil}{2^q} - 1\right). \tag{3.1.83}$$

Now, s is a uniform random variable getting the 2^q values of,

$$s \in \left\{2^{1-q}kA - A \middle| k = 1 \cdots 2^q\right\}, \tag{3.1.84}$$

with the same probability. Here, q is the number of quantization steps of s and A is its maximum amplitude. Now, assume computing,

$$k = 2^{q-1}\left(1 + \frac{s}{A}\right). \tag{3.1.85}$$

Simple manipulations show that k is a q–bit unsigned integer.

Thus, having a random generator giving a uniform random variable in the range $(0, 1]$, we propose to use (3.1.83) to have a new uniform random variable in the range $(-A, A]$. The importance of the new random variable is that its values are represented in q–bits in a lossless manner. In contrast, for near lossless storage of the original random variable, one should spare 64–bit of memory for storing a double precision floating point number.

3.1.9 Quality Measures

The classic measure of quality is the *peak signal to noise ratio* (PSNR), computed as,

$$PSNR = 20 \log_{10} \frac{255}{\sqrt{E\{(I - \tilde{I})^2\}}}, \tag{3.1.86}$$

where, I is the original signal and $\tilde{I}$ is the degraded one. Here, it is assumed that the dynamic range of both I and $\tilde{I}$ are $[0, 255]$. A similar measure is the *signal to noise ratio* (SNR) defined as,

$$SNR = 10 \log_{10} \frac{E\{I^2\}}{E\{(I - \tilde{I})^2\}}, \tag{3.1.87}$$

Added to these measures, we use another measure, called maximum deviation, defined as,

$$Max.\ Dev. = 20 \log_{10} \frac{255}{\sqrt{\max\{(I - \tilde{I})^2\}}}. \tag{3.1.88}$$

While maximum deviation is always less than (rarely equal to) PSNR, it reveals the worst degradation of the input signal. In this way, there is a possibility to have a

psnr

```
function p=PSNR(X,Y,Maximum)
%Function PSNR computed the PSNR of two same-sized matrices X and Y.
%Usage: p=PSNR(X,Y,Maximum)
%Inputs:     X,Y:Input matrices.
%           Maximum:Use maximum deviation (default : 0)
%Outputs:    p:PSNR.
%Revision 1.1 Arash Abadpour Winter 2005 (March 3, 3005)
```

Figure 3.14: Implementation of the *psnr* function.

signal with high PSNR, while it contains very small portions which are massively degraded. For these signals the fall of maximum deviation will be a helpful indicator.

To compute PSNR and SNR, one may use the functions *psnr* and *snr* in **CIP**. Both functions give two same–sized arrays with an arbitrary dimension (see Figure 3.14 and 3.15). The function *psnr* alsocomputed Max. Dev. when the argument *maximum* is unity.

3.1.10 Finding the Closest Point to a set of Lines.

Assume that n lines of $l_1, \cdots, l_n$in R^m are given. Also, assume that l_i is indicated by the two vectors $\vec{\eta_i}$ and $\vec{v_i}$. In this way, $\vec{\eta_i}$ is a vector starting from the origin, with its head on l_i and $\vec{v_i}$ is the direction of l_i, satisfying $\|\vec{v_i}\| = 1$. Hence, the points on l_i occupy $L_i \subset R^m$, which is defined as,

$$L_i = \{\vec{\eta_i} + \lambda\vec{v_i} | \lambda \in R\}. \tag{3.1.89}$$

snr

```
function p=SNR(X,Y,Maximum)
%Function SNR computed the SNR of two same-sized matrices X and Y.
%Usage: p=SNR(X,Y)
%Inputs:     X:Original signal.
%            Y:Noisy signal.
%            Maximum:Use maximum deviation (default : 0)
%Outputs:    p:SNR.
%Revision 1.1 Arash Abadpour Spring 2005 (April 4, 2004)
```

Figure 3.15: Implementation of the *snr* function.

We assume that $\vec{\eta}_i$ is perpendicular to $\vec{v}_i$ ($\vec{\eta}_i^T \vec{v}_i = 0$). If this condition is not met, $\vec{\eta}_i$ is replaced with,

$$\vec{\eta}_i^* = \vec{\eta}_i - (\vec{v}_i^t \vec{\eta}_i)\vec{v}_i. \tag{3.1.90}$$

The closest point to the n lines $l_1, \cdots, l_n$ is the point $\vec{x} \in R^m$ minimizing the following objective function,

$$\Delta = \sum_{i=1}^{n} d(\vec{x}, l_i)^2, \tag{3.1.91}$$

where, $d(\vec{x}, l_i)$ is the distance from the point $\vec{x}$ to the line l_i. Linear algebra shows that,

$$d(\vec{x}, l_i)^2 = \left\| (\vec{x} - \vec{\eta}_i) - \left[\vec{v}_i^T (\vec{x} - \vec{\eta}_i)\right] \vec{v}_i \right\|^2, \tag{3.1.92}$$

which is simplified to,

$$d(\vec{x}, l_i)^2 = (\vec{x} - \vec{\eta}_i)^T (\vec{x} - \vec{\eta}_i) - (\vec{x} - \vec{\eta}_i)^T \vec{v}_i \vec{v}_i^T (\vec{x} - \vec{\eta}_i), \tag{3.1.93}$$

$$d(\vec{x}, l_i)^2 = (\vec{x} - \vec{\eta}_i)^T \left[I - \vec{v}_i \vec{v}_i^T\right](\vec{x} - \vec{\eta}_i). \tag{3.1.94}$$

Substituting (3.1.94) in (3.1.91) gives,

$$\Delta = \sum_{i=1}^{n} (\vec{x} - \vec{\eta}_i)^T \left[I - \vec{v}_i \vec{v}_i^T\right](\vec{x} - \vec{\eta}_i). \tag{3.1.95}$$

From calculus we know that [182],

$$\frac{\partial(\vec{x}^T A \vec{x})}{\partial \vec{x}} = A\vec{x} + A^T \vec{x}, \tag{3.1.96}$$

Hence,

$$\frac{\partial \Delta}{\partial \vec{x}} = 2 \sum_{i=1}^{n} \left[I - \vec{v}_i \vec{v}_i^T\right](\vec{x} - \vec{\eta}_i). \tag{3.1.97}$$

Setting $\frac{\partial \Delta}{\partial \vec{x}} = 0$ gives,

$$\left[I - \frac{1}{n} \sum_{i=1}^{n} \vec{v}_i \vec{v}_i^T\right] \vec{x} = \frac{1}{n} \sum_{i=1}^{n} \vec{\eta}_i. \tag{3.1.98}$$

Lets spend some time to prove that the $m \times m$ matrix in left (lets call it U) is always non–singular. A theory in linear algebra states that: "*if $I + E$ is singular then $\|E\| \geq 1$, where $\|\cdot\|$ is a subordinate matrix norm*" [151]. We will first prove this theorem. Assume that $I + E$ is singular. Hence, there exists a non–zero vector $\vec{x}$ satisfying $(I + E)\vec{x} = \vec{0}$. Thus, $\vec{x} = -E\vec{x}$ or equivalently,

$$\|E\| \geq \frac{\|E\vec{x}\|}{\|\vec{x}\|} = 1. \tag{3.1.99}$$

which proves the theorem.

Now, to prove that U in (3.1.98) is non–singular, it suffices to show that,

$$\| \sum_{i=1}^{n} \vec{v}_i \vec{v}_i^T \| \leq n. \tag{3.1.100}$$

We prove (3.1.100) by showing that for any vector $\vec{y}$ satisfying $\|\vec{y}\| = 1$,

$$\|\sum_{i=1}^{n} \vec{v}_i \vec{v}_i^T \vec{y}\| < n, \tag{3.1.101}$$

First, we know that for any two arbitrary $m \times 1$ vectors $\vec{u}$ and $\vec{v}$, $|\vec{u}^T\vec{v}| \leq \sqrt{\|\vec{u}\|^2\|\vec{v}\|^2}$. Note that equality occurs for $\vec{u} \| \vec{v}$. Thus, in the current case we have $|\vec{v}_i^T \vec{y}| \leq 1$. Now,

$$\|\sum_{i=1}^{n} \vec{v}_i \vec{v}_i^T \vec{y}\| \leq \sum_{i=1}^{n} |\vec{v}_i^T \vec{y}| \|\vec{v}_i\| \leq \sum_{i=1}^{n} \|\vec{v}_i\| = n, \tag{3.1.102}$$

As such, (3.1.101) is proved if there exists i and j satisfying $\vec{v}_i$ is not parallel to $\vec{v}_j$. Neglecting this odd situation, it is proved that (3.1.98) always has a solution as,

$$\vec{x} = \Big[nI - \sum_{i=1}^{n} \vec{v}_i \vec{v}_i^T\Big]^{-1} \sum_{i=1}^{n} \vec{\eta}_i. \tag{3.1.103}$$

Now, $\vec{x}$ is the closest point to the n lines $l_1, \cdots, l_n$.

3.1.11 Finding the Closest Point to a set of Fuzzy Lines.

Assume that we are seeking the closes point to the set of n lines $l_1, \cdots, l_n$ in R^m. Also, assume that the importance of l_i is p_i. As such, we are looking for $\vec{x} \in R^m$ which minimize,

$$\tilde{\Delta} = \sum_{i=1}^{n} p_i d(\vec{x}, l_i)^2, p_i > 0, \tag{3.1.104}$$

Note that setting $\forall i, p_i = 1$ converts (3.1.104) into (3.1.91). An analysis similar to the one performed in Section 3.1.10 reveals that $\vec{x}$ is computed as,

$$\vec{x} = \Big[\sum_{i=1}^{n} p_i I - \sum_{i=1}^{n} p_i \vec{v}_i \vec{v}_i^T\Big]^{-1} \sum_{i=1}^{n} p_i \vec{\eta}_i. \tag{3.1.105}$$

Again note that assuming $\forall i, p_i = 1$ converts (3.1.105) into (3.1.103), as expected.

3.2 Spotting

In this section, a color spotting task is investigated. The investigation includes utilizing thirteen color spaces (see Section 2.1) and two proposed 1–D color descriptors. In the spotting problem we assume that a query region r is given. The solution to the problem is to give a discriminating function f which separates color vector similar to r and those different from it in the entire image.

3.2.1 Color Spaces

Although, any color space is defined as a function $G : R^3 \rightarrow R^3$, here we face each channel of a color space independently. Thus, we are concerned to perform the classification according to a function $X : R^3 \rightarrow R$. The discrimination function is defined as:

$$f_r^X(\vec{c}) = \begin{cases} 1, |X(\vec{c}) - C| \leq T \\ 0, otherwise \end{cases} , \tag{3.2.106}$$

where $X(\cdot)$ is a function producing one of the channels of a selected color space out of the coordinates of $\vec{c}$ in RGB space. Also, C and T denote the center and threshold of color cluster. These two parameters control the spread of discriminated members. These two parameters can be either left to the user or computed using the statistics of r.

Note that the loci of vectors highlighted by the discriminator function defined in (3.2.106) is a blurred version of a 2–D space. Here, the 2–D skeleton is the loci of all

color vectors satisfying $X(\vec{c}) = C$. This skeleton may be a plane (*e.g.*, I_3), cylinder (*e.g.*, $HSI.S$), pie (*e.g.*, $HSV.H$), or many other shapes (see Figure 3.16).

3.2.2 Parametric Linear Adaptive Color (PLAC) Descriptor

Most of the color spaces suffer from the disadvantageous fixed structures that makes them inefficient in treating special odd–shaped loci in the color space. This is the main motivation for defining the *parametric linear adaptive color* (PLAC) descriptor. The general idea is to decide the membership status using a linear combination of RGB coordinates:

$$|a_r R + a_g G + a_b B - C| \leq T \geq 0, \tag{3.2.107}$$

where a_r, a_g, a_b, C, and T are controlled by the user. Due to the correlation between a_r, a_g and a_b parameters and T, the formulation in (3.2.107) is modified. First consider the function f for fixed values of a_r, a_g, and a_b,

$$f(R, G, B) = a_r R + a_g G + a_b B. \tag{3.2.108}$$

Now assume finding the extremes of f. As R, G, and B accept values in the range of $[0, 255]$, the minimum value of $f(R, G, B)$ occurs when those with negative multipliers accept the highest value (255),

$$\min_{R,G,B \in [0,255]} f(R, G, B) = 255 \sum_{a_x < 0} a_x. \tag{3.2.109}$$

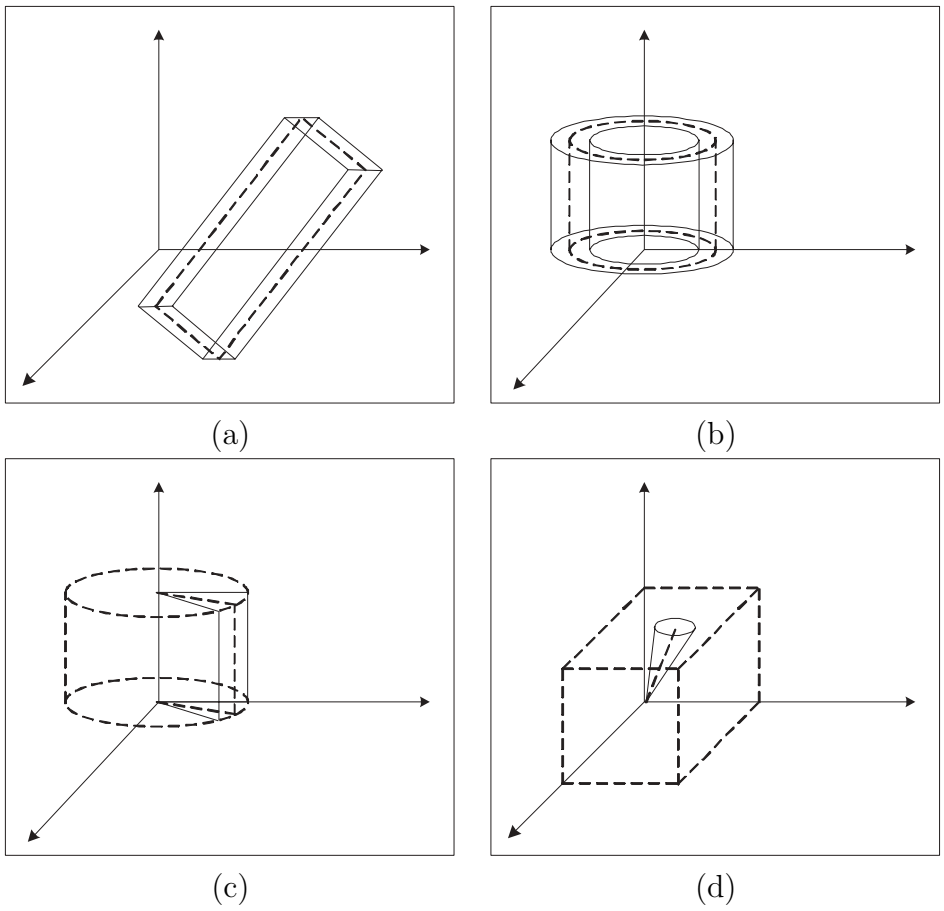


Figure 3.16: Possible highlighted loci using color spaces. In each case solid lines show the borders of the locus and dashed lines illustrate the skeleton. (a) Planar. (b) Cylindrical. (c) Angular. (d) Possible shape.

Similarly, we have,

$$\max_{R,G,B\in[0,255]} f(R,G,B) = 255 \sum_{a_x>0} a_x. \tag{3.2.110}$$

Thus,

$$255 \sum_{a_x<0} a_x \leq a_r R + a_g G + a_b B \leq 255 \sum_{a_x>0} a_x, \tag{3.2.111}$$

$$0 \leq a_r R + a_g G + a_b B - 255 \sum_{a_x<0} a_x \leq 255 \sum_{a_x>0} a_x - 255 \sum_{a_x<0} a_x, \tag{3.2.112}$$

$$0 \leq a_r R + a_g G + a_b B - 255 \sum_{a_x<0} a_x \leq 255 \sum |a_x|, \tag{3.2.113}$$

$$0 \leq \frac{a_r}{255 \sum |a_x|} R + \frac{a_g}{255 \sum |a_x|} G + \frac{a_b}{255 \sum |a_x|} B - \frac{\sum_{a_x<0} a_x}{\sum |a_x|} \leq 1. \tag{3.2.114}$$

Now its reasonable to control the spread of the middle term using thresholding,

$$\Big| \frac{a_r}{255 \sum |a_x|} R + \frac{a_g}{255 \sum |a_x|} G + \frac{a_b}{255 \sum |a_x|} B - \frac{\sum_{a_x<0} a_x}{\sum |a_x|} - \frac{C}{255} \Big| \leq \frac{T}{255}. \tag{3.2.115}$$

Here, both C and T are in the range of $[0, 255]$. Thus,

$$\Big| a_r R + a_g G + a_b B - 255 \sum_{a_x<0} a_x - C \sum |a_x| \Big| \leq T \sum |a_x|, \tag{3.2.116}$$

$$\left| a_r R + a_g G + a_b B - 255 \Big[\sum_{a_x<0} a_x + C \frac{\sum |a_x|}{255} \Big] \right| \leq T \sum |a_x|, \tag{3.2.117}$$

$$\left| a_r R + a_g G + a_b B - \tilde{C} \right| \leq \tilde{T}, \tag{3.2.118}$$

where,

$$\tilde{C} = 255\Big[\sum_{a_x<0} a_x + C\frac{\sum|a_x|}{255}\Big], \tilde{T} = T\sum|a_x|. \tag{3.2.119}$$

A little manipulation leads to the PLAC discriminator function defined as,

$$\begin{cases} f^{PLAC}_{a_r,a_g,a_b,C,T}(\vec{c}) = \begin{cases} 1, |\vec{a}^T\vec{c} - \tilde{C}| \leq \tilde{T} \\ 0, otherwise \end{cases} \\ \tilde{C} = 255\Sigma_{a_x<0}a_x + C\Sigma|a_x|, \tilde{T} = T\Sigma|a_x| \\ \vec{a} = \begin{bmatrix} a_r & a_g & a_b \end{bmatrix}^T \end{cases}. \tag{3.2.120}$$

PLAC is a $1-D$ color descriptor; in contrast with the ordinary $3-D$ and $4-D$ color spaces. Note that there are five user–tuned parameters in PLAC. Also, note that like the case of the color spaces, the loci of color vectors highlighted by the PLAC has a 2–D skeleton. In contrast with the varying shape of skeletons in the color spaces, the skeleton of the PLAC loci is always a plane.

3.2.3 PCA–based Parametric Linear Adaptive Color Descriptor

As the tuning phase of PLAC depends on user, a new color descriptor named as PCA–based parametric linear adaptive color (PCA–PLAC) descriptor is also introduced.

Rather than the numerical parameters tuned by the user in PLAC and other color descriptors, PCA–PLAC extracts the information from the scene. When trying to use PCA–PLAC, one must give a rectangle of the desired segment to the algorithm (Let's call the region as r.). By forming the $3 \times A$ (A is the area of r) matrix S containing

the RGB values of all pixels in r , the vector $\vec{\eta}$ is computed by row averaging of S to give $E_{\vec{c}\in r}\{\vec{c}\}$ as a 3×1 vector. This vector is used to produce the matrix D as the center oriented version of S . The eigenvalues of the matrix $C = D^T D$ are computed and the eigenvector corresponding to the largest eigenvalue is selected to be $\vec{v}$.

The *reconstruction error* (RE) of a point regarding to the region r is defined as (3.2.121) in which smaller values shows more tendency. In (3.2.121), $\langle \vec{v} \rangle$ is a custom norm function defined in (3.2.122). The marginal reconstruction error is computed as (3.2.123) and a tolerance is asked from the user (λ). The classification function for an arbitrary point is defined in (3.2.124).

$$\tau_r(\vec{c}) = \langle \acute{\vec{v}}[\vec{c} - \vec{\eta}]\vec{v} - [\vec{c} - \vec{\eta}] \rangle, \tag{3.2.121}$$

$$\langle \vec{v} \rangle = \frac{1}{3}\Big(|v_1| + |v_2| + |v_3|\Big), \tag{3.2.122}$$

$$\tilde{\tau}_r = \arg_{c\in r}\{P(\tau_r(\vec{c}) < e) > 0.95\}, \tag{3.2.123}$$

$$f_{r,\lambda}^{PCA--PLAC}(\vec{c}) = \begin{cases} 1, \tau_r(\vec{c}) \leq \lambda\tilde{\tau}_r \\ 0, otherwise \end{cases} \quad . \tag{3.2.124}$$

It must be emphasized that although PCA–PLAC needs user to draw a rectangle on the selected object, there is only one user–selected parameter to be tuned in PCA–PLAC in contrast with the five parameters in PLAC. It is worth mentioning that tuning PCA–PLAC is more intuitive compared to tuning PLAC.

To find out the repeatability of PCA–PLAC, the correlation between different results of spotting one object is computed as (3.2.125), along with a parameter showing the range of the tolerance in different tests on the same object as (3.2.126).

$$\delta_{I_1 I_2} = 2\frac{E\{I_1 I_2\}}{E\{I_1\} + E\{I_2\}}, \tag{3.2.125}$$

$$\tilde{\delta}_\lambda = \frac{\sigma_\lambda}{E\{\lambda\}}. \tag{3.2.126}$$

3.2.4 Comparison Method

The 13 different color spaces under investigation are RGB, $CMYK$, HSI, $I_1I_2I_3$, $CIE-Lab$, $CIE-LHC$, $CIE-Luv$, $CIE-XYZ$, YC_bC_r, YIQ, YUV, and $HMMD$. According to the categorizes of color spaces declared in section 2.1, There are four HVS–based, four application–specific, and four CIE color spaces involved in this study.

The objects in the sample image (see Figure 3.17) were indexed and their areas were calculated by manual segmentation with repeatedly use of *magic select* tool in *Adobe Photoshop*. To analyze the performance of spotting in the pre–described color spaces, after computing the representation of the sample image in 13 color spaces (41 channels), answers to the following questions were inspected in each of the 41 channels for each of the 8 objects:

1. How much is the most percentile of the object area when it is cut out of the image with the best–selected set of parameters? ($Q_1 \in [0 \dots 100]$)

Figure 3.17: An image taken from eight objects with different colors.

2. When trying to answer the first question, is the background completely removed, without considerable intrusion? ($Q_2 \in \{0, 100\}$.)

Also, a zero–one fail rate parameter is defined; to show the situations where the method is unable to distinguish the border of the object. The answers to these questions were acquired and statistically analyzed.

Using a *graphic user interfaces* (GUI) developed in MATLAB 6.5 with scroll bars for tuning parameters (C,T in color spaces, a_r, a_g, a_b, C, and T in PLAC and λ in PCA–PLAC). He was looking at the original image and the spotted image in two aside windows.

3.3 Distance Functions

Assume that the set of 3–D color vector $r = \{\vec{c}_1, \cdots \vec{c}_N\}$ and the arbitrary color vector $\vec{c}$ are given. We call a function $\tau_r : R^3 \rightarrow R^+ \cup \{0\}$ as a distance function for r, if it satisfies:

$$\forall \vec{c}_1 \in r, \forall \vec{c}_2 \in R^3 - r, \tau_r(\vec{c}_1) \preceq \tau_r(\vec{c}_2). \tag{3.3.127}$$

Here, $x \preceq y$ means that x is mainly smaller than y. Roughly speaking, τ_r should rank the members of r with relatively smaller values, compared to the other vectors.

Many different kinds of distance functions are available in the literature. The two most famous are Euclidean and Mahalanobis distances. The Euclidean distance of the color vector vector $\vec{c}$ to the color cluster r is defined as,

$$\tau_r^E(\vec{c}) = \|\vec{c} - \vec{\eta}_r\|^2, \tag{3.3.128}$$

where $\vec{\eta}_r$ is the expectation of the members of cluster r.

$$\vec{\eta}_r = \frac{1}{N} \sum_{i=1}^{N} \vec{x}_i. \tag{3.3.129}$$

Also, the Mahalanobis distance of the color vector $\vec{c}$ to the color cluster r is defined as,

$$\tau_r^M(\vec{c}) = (\vec{c} - \vec{\eta}_r)^T C_r^{-1} (\vec{c} - \vec{\eta}_r). \tag{3.3.130}$$

Here, $\vec{\eta}_r$ is the expectation vector of the members of r and C_r is the covariance matrix

of r, defined as,

$$C_r = \frac{1}{N}\sum_{i=1}^{N}(\vec{x}_i - \vec{\eta}_r)(\vec{x}_i - \vec{\eta}_r)^T. \tag{3.3.131}$$

Furthermore, the *linear partial reconstruction error* (LPRE) distance of the vector $\vec{c}$ to the cluster r is defined as,

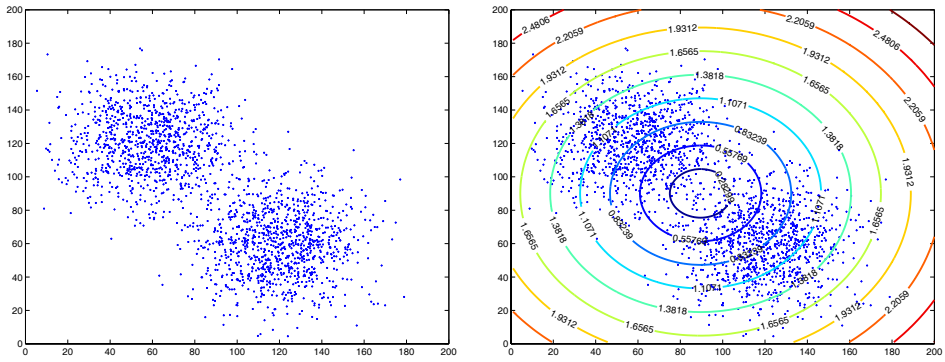
$$\tau_r^L(\vec{c}) = \|\vec{v}^T(\vec{c} - \vec{\eta}_r)\vec{v}_r - (\vec{c} - \vec{\eta}_r)\|^2, \tag{3.3.132}$$

where, $\vec{v}_r$ and $\vec{\eta}_r$ show the direction of the first principal component and the expectation vector of the cluster r. Then,

$$\vec{v}_r = \arg_{\|\vec{\omega}\|=1, C_r\vec{\omega}\|\vec{\omega}} \left\{ \frac{\|C_r\vec{\omega}\|}{\|\vec{\omega}\|} \right\}. \tag{3.3.133}$$

Here, $\|\vec{x}\|$ denotes the L_2 norm. Also, using the normalized L_1 norm yields the same results.

Reviewing the cluster shapes corresponding to different distance functions is a keen way to adapt the method with the intrinsic characteristics of the problem. Here, we will briefly review the geometrical implications of these three distance functions. Although, any distance function could be defined for any color cluster, one should be aware of using a distance function which is based on a non–homogenous set. For example assume the set of 2–D points shown in Figure 3.18–a as r. Figure 3.18–b shows the corresponding distribution of the Euclidean distance values. Note the large set of members not included in r but giving low values of distance. This effect



(a) (b)

Figure 3.18: (a) A non–homogenous set. (b) Distribution of Euclidean distances.

mainly corresponds to the non–homogeneity (here, non–compactness) of the original set. Thus, the geometry of iso–curves (the loci of points with the same distance) of a distance function completely depends on the assumptions about the spread of the points of the phenomenon under investigation.

The Euclidean distance function only depends on the mean vector of the color cluster. As a result, each homogenous region is modelled as a sphere. In the same way, the Mahalanobis distance is presuming clusters to have elliptical shapes and the LPRE defines cylindrical clusters. Note that the iso–ellipses produced by the Mahalanobis distance are not aligned on the xyz coordinate axes. In fact, the ellipse is rotated by the PCA matrix of the color vectors in r. The diagonals of the ellipse relate to the eigenvalues of the covariance matrix. The LPRE loci only depends on the first eigenvector of the PCA matrix. This vector defines the main axis of the data cloud. Figure 3.19 illustrates these cluster shapes in the 2–D domain.

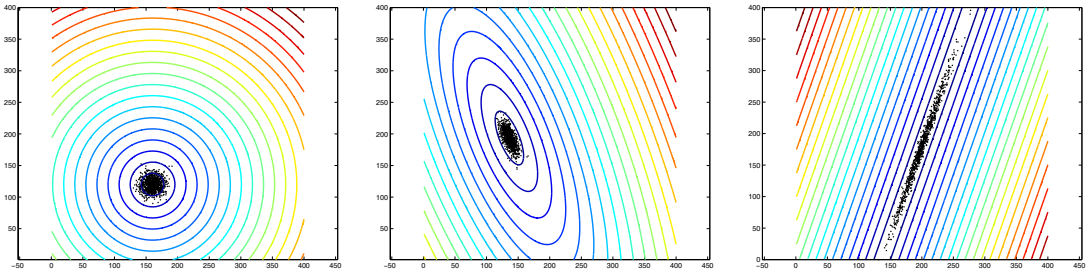


Figure 3.19: (a) Spherical cluster. (b) Elliptical cluster. (c) Cylindrical cluster.

Although, the definition of these distance functions correspond to the crisp domain, we can further extend them to the fuzzy domain. In this approach, the fuzzy cluster $\tilde{r}$ is defined as a fuzzy set containing $\vec{x}_i$ with the membership value of p_i,

$$\tilde{r} = \{(\vec{x}_i; p_i) | i = 1, \cdots, N\}. \tag{3.3.134}$$

Now, the fuzzy expectation, the fuzzy covariance matrix, and the fuzzy first principal component direction are defined as,

$$\tilde{\vec{\eta}}_r = \frac{1}{\sum_{i=1}^{N} p_i} \sum_{i=1}^{N} p_i \vec{x}_i, \tag{3.3.135}$$

$$\tilde{C}_r = \frac{1}{\sum_{i=1}^{N} p_i} \sum_{i=1}^{N} p_i (\vec{x}_i - \vec{\eta}_r)(\vec{x}_i - \vec{\eta}_r)^T, \tag{3.3.136}$$

$$\tilde{\vec{v}} = \arg_{\|\vec{\omega}\|=1, \tilde{C}_r \vec{\omega} \| \vec{\omega}} \left\{ \frac{\|\tilde{C}_r \vec{\omega}\|}{\|\vec{\omega}\|} \right\}. \tag{3.3.137}$$

In Section 3.1.1 we proved that these definitions are mathematically sound. Note that reducing the fuzzy set $\tilde{r}$ to the crisp set r, by setting all p_i equal to unity, converts all

fuzzy notations to their corresponding crisp definitions. Thus, we can meaningfully talk about the fuzzy Euclidean, Mahalanobis, and LPRE distances as,

$$\tau_{\tilde{r}}^{E}(\vec{c}) = \|\vec{c} - \vec{\tilde{\eta}_{\tilde{r}}}\|^2, \tag{3.3.138}$$

$$\tilde{\tau}_{\tilde{r}}^{M}(\vec{c}) = (\vec{c} - \vec{\tilde{\eta}_{\tilde{r}}})^T \tilde{C}_{\tilde{r}}^{-1} (\vec{c} - \vec{\tilde{\eta}_{\tilde{r}}}), \tag{3.3.139}$$

$$\tau_{\tilde{r}}^{L}(\vec{c}) = \|\vec{\tilde{v}_{\tilde{r}}^T}(\vec{c} - \vec{\tilde{\eta}_{\tilde{r}}})\vec{\tilde{v}_{\tilde{r}}} - (\vec{c} - \vec{\tilde{\eta}_{\tilde{r}}})\|^2. \tag{3.3.140}$$

In the foregoing part of this thesis, we assume all sets to be fuzzy. Hence, without mentioning that the sets are either fuzzy or crisp we use fuzzy domain tools (setting $p_i \equiv 1$ when appropriate).

Use the following stochastic margin to compute the homogeneity of the selected cluster r,

$$\|f\|_{r,p} = \tag{3.3.141}$$

$$\arg_e\{P_{\vec{x}\in r}\{f(\vec{x}) \leq e\} \geq p\},$$

where p is the inclusion percentage and $P_{\vec{x}\in r}\{f(\vec{x}) \leq e\}$ denotes the probability of x being less than or equal to e. Equation (3.3.141) leads to the definition of the normalized LPRE measure as,

$$\tilde{\tau}_{r,p}(\vec{c}) = \frac{\tau_r(\vec{c})}{\|\tau_r\|_{r,p}}. \tag{3.3.142}$$

Regarding the definition of the normalized reconstruction error in (**??**) and the reformulated Butterworth function in (3.1.68), an image is fuzzificated regarding the

PCAImembership
function J=PCAIMembership(I,R,P,kind,dxdy)
%Function PCAImage Membership computes the membership of image points
%regarding the region R.
%Usage: J=PCAIMembership(I,R,P,kind)
%Inputs: I:Input image
% R:Region of the form [y1,x1,y2,x2]
% P:Inclusion Percentage
% kind:0:Euclidean, 1:Mahalanobis, 2:Reconstruction Error.
% dxdy:Averaging kernel radius.
%Output: J:Membership
%Revision 1.22 Arash Abadpour Winter 2005 (March 18, 2005)

Figure 3.20: Implementation of the *PCAImembership* function.

query region r by using $B_{\alpha,\beta}\big(\tilde{\tau}_{r,p}(\vec{c})\big)$. Note that the points in r (and points similar to them in color sense) are mostly giving membership values in the range of $[1, \alpha]$. The color vectors that are not like those existing in r are ranked with poor values. The proposed fuzzification scheme is performed by the two functions *PCAImembership* and *PCAImembershipmask* (see Figures 3.20 and 3.21).

The comprehensive investigation of the above fuzzification method shows its performance for processing of color images (see Section 4.3). Also, it is shown that setting $p = \frac{1}{2}$ is a proper choice in all conditions. The values of α and β control the shape of the membership function and are left to the user to tune the function to well fit the constraints of the problem in hand. As for all $x \in [0, 1]$, $B_{\alpha,\beta}(x)$ must gain high levels of membership, using $\alpha = 0.99$ is a reasonable choice. Now, By tuning the β parameter, one can easily control the distribution of the membership function. This

PCAImembershipMask
function J=PCAIMembershipMask(I,M,P,kind)
%Function PCAImage Membership computes the membership of image points
%regarding the region R.
%Usage: J=PCAIMembershipMask(I,M,P,kind)
%Inputs: I:Input image
% M:Mask
% P:Inclusion Percentage
% kind:0:Euclidean, 1:Mahalanobis, 2:Reconstruction Error.
%Output: J:Membership
%Revision 1.0 Arash Abadpour Summer 2004 (1383/5/14)

Figure 3.21: Implementation of the *PCAImembershipMask* function.

leads to the single parameter fuzzification function defined as,

$$h_{r,\beta}(\vec{c}) = B_{0.99,\beta}\big(\tilde{\tau}_{r,\frac{1}{2}}(\vec{c})\big). \tag{3.3.143}$$

Figure 3.22 shows the results of the proposed fuzzification method on the sample image *Peppers*. Corresponding β values are denoted in the caption of Figure 3.22. Here, the query region is a rectangular part of the red pepper (see Figure 3.22–a).

3.4 Query Region Extraction

In this section two query region extract methods are proposed. The LPRE–based method is a fast and precise method discussed in Section 3.4.1, while the LLPRE method is a more accurate approach (see Section 3.4.2).

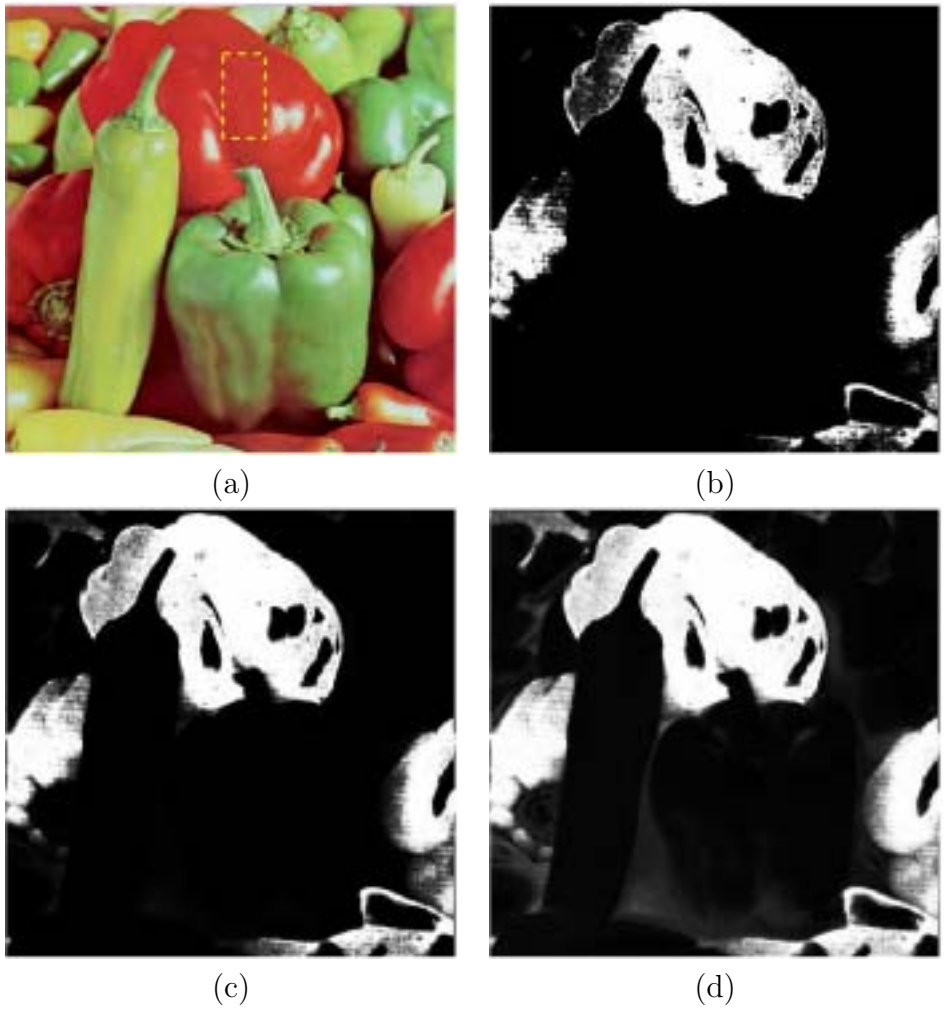


Figure 3.22: Fuzzification of *Peppers* according to the shown query region shown with different values of β. (a) Original image overlayed the query region, (b) $\beta = 0.1$, (c) $\beta = 0.6$, and (d) $\beta = 0.9$.

3.4.1 LPRE–Based Approach

Assume that the image I and the query region r (within I) are given. The goal is to find the binary image M containing one connected component with smooth edges and few holes. The mask image is desired to highlight a maximal homogenous region in I, containing large portion of the query region.

Assume that we have intentionally selected one of the normalized likelihood measures discussed in Section 3.3. As shown in Figure 3.23, using the query region and the likelihood function, $\tilde{\tau}^{\star}_{r,p}$, the given image is first converted to a likelihood map showing the likelihood of each pixel to the query region r. Here, the parameter p rules the margins of the resulting segment. In fact, p controls the inclusion of the outliers in the color scheme, yielding a limit on the margins of the selected color content. To incorporate the spatial information into the likelihood map, it is then fed into a conventional low–pass filter to give a segment with smoothed edges. To avoid the time consuming Fourier domain operations, we propose to use spatial domain averaging using a disk with radius ρ. Holding the points with values of the filtered likelihood lower than unity and rejecting others, a binary map is obtained. This map has smooth edges, because of the utilized spatial filtering process. Also, it contains no holes with radius lower than ρ, because such holes are filled by the filtering process. As described above, the mask should contain only one connected component, thus, the produced map is decomposed into its connected components and the one

correlating more with the query region, is selected.

Assume that for the given query region, r, we have manually computed the desired mask, call it $M_\circ$. Also, assume that a query region extraction process has resulted in M. We propose two measures to investigate the performance of such methods. These two measures compare M and $M_\circ$ in terms of recall and precision, denoted by μ_r and μ_p, respectively.

$$\mu_r = \frac{\|M \cap M_\circ\|}{\|M_\circ\|}, \tag{3.4.144}$$

$$\mu_p = \frac{\|M \cap M_\circ\|}{\|M\|}, \tag{3.4.145}$$

where, $\|A\|$ is the cardinality of the set A defined as the number of its members, (here interpreted as its area). While, μ_r computes the intersection between M and $M_\circ$, evaluating the desired area highlighted by the method, μ_p focuses on the amount of outlier points. Thus, a query region extraction method is desirable if it results in $\mu_r \simeq 1$ and $\mu_p \simeq 1$. Note that the above defined measures are both in the range of $[0, 1]$. Functions *QuerySegment*, *SegSim*, and *SegSimMask* are designed to perform query region extraction.

3.4.2 LLPRE–Based Approach

Figure 3.27 shows a sample query region extraction result using LPRE. Although, outperforming other available color vector distance functions, LPRE incorporates certain type of malfunction which is related to its structure. Figure 3.29–a illustrates LPRE distance of vectors to a sample cluster. Note that each LPRE cluster is defined

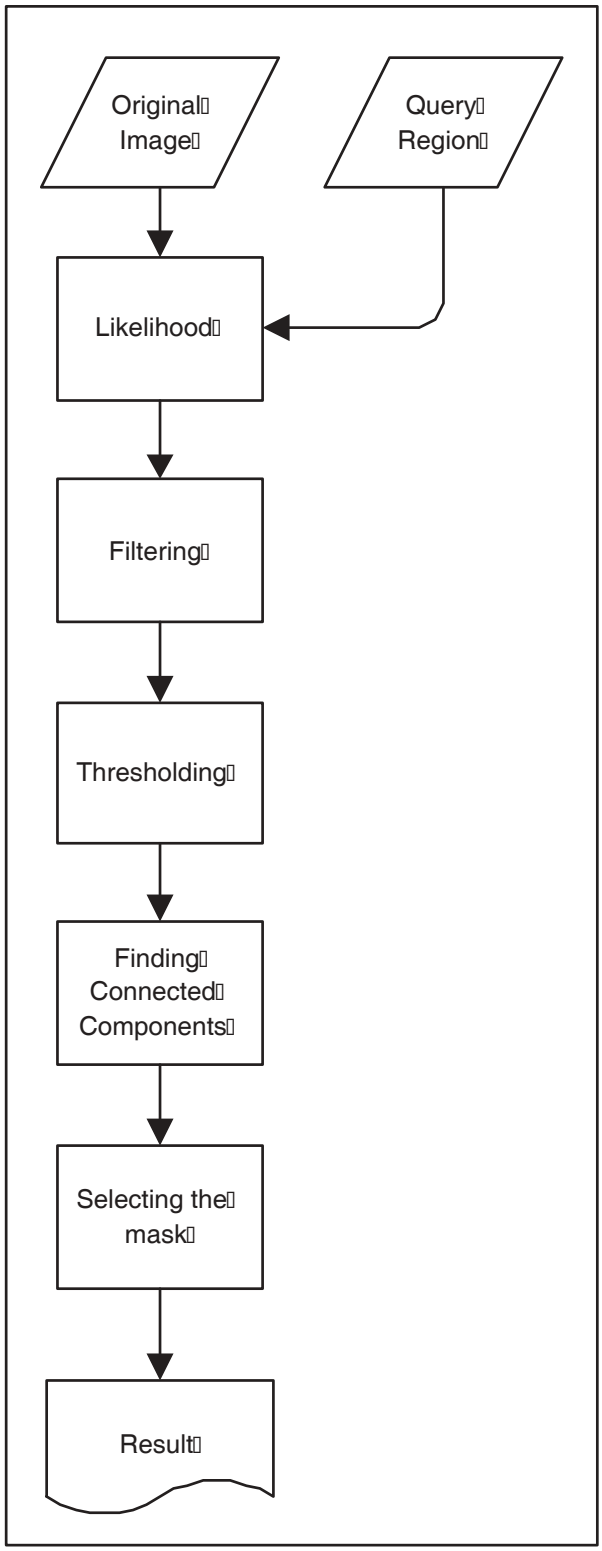


Figure 3.23: Flowchart of the proposed query region extraction.

SegSim

```
function [S,J,F]=SegSim(I,R,dxdy,percentage,singleregion,kind,alpha,beta)
%The Segment Similar segments the similar region in the image I.
%Usage:[S,J,F]=SegSim(I,R,dxdy,percentage,singleregion,kind,alpha,beta)
%Inputs:     I:Original image.
%           R:Base region.
%           alpha,beta,percentage: Fuzzification parameters
%           dxdy:Lowpass filter radius.
%           threshold:Plane height
%           singleregion:Find a single region?
%           kind:0:Euclidean, 1:Mahalanobis, 2:Reconstruction Error.
%           alpha,beta:Butterworth setting.
%
%Outputs:    S:Snake as a 2*N array.
%                   J:Membership values.
%                   F:Fuzzy membership.
%Revision 2.21 Arash Abadpour Winter 2005  (March 18, 2005)
```

Figure 3.24: Implementation of the *SegSim* function.

SegSimMask

```
function [S,J,F]=SegSimMask(I,M,dxdy,percentage,singleregion,kind,alpha,beta)
%The Segment Similar segments the similar region in the image I.
%Usage:[S,J,F]=SegSimMask(I,M,dxdy,percentage,singleregion,kind,alpha,beta)
%Inputs:     I:Original image.
%           M:Mask.
%           alpha,beta,percentage: Fuzzification parameters
%           dxdy:Lowpass filter spatial frequency
%           threshold:Plane height
%           singleregion:Find a single region?
%           kind:0:Euclidean, 1:Mahalanobis, 2:Reconstruction Error.
%           alpha,beta:Butterworth setting.
%
%Outputs:    S:Snake as a 2*N array.
%                   J:Membership values.
%                   F:Fuzzy membership.
%Revision 1.0 Arash Abadpour Summer 2004 (1383/5/14)
```

Figure 3.25: Implementation of the *SegSimMask* function.

QuerySegment

```
function [S,J,F]=
%       QuerySegment(I,R,dxdy,percentage,singleregion,kind,cs,ch,alpha,beta)
%The query segment function segments the region similar to the query region
%using specified component of the speciefied colro space or uses
%Mahalanobis/LPRE likelihood measures.
%Usage:[S,J,F]=
%         QuerySegment(I,R,dxdy,percentage,singleregion,kind,cs,ch,alpha,beta)
%Inputs:    I:Original image.
%          R:Base region.
%          alpha,beta,percentage: Fuzzification parameters
%          dxdy:Lowpass filter spatial frequency
%          threshold:Plane height
%          singleregion:Find a single region?
%          kind:0:Euclidean, 1:Mahalanobis, 2:Reconstruction Error.
%          cs:Color space.
%          ch:color channels.
%          alpha,beta:Butterworth setting.
%
%Outputs:   S:Snake as a 2*N array.
%          J:Membership values.
%          F:Fuzzy membership.
%Revision 1.01 Arash Abadpour Summer 2004 (1383/5/17)
```

Figure 3.26: Implementation of the *QuerySegment* function.

(a) (b)

Figure 3.27: Query region extraction results using LPRE. (a) Original image and the query region. (b) Extracted region.

as a line, thus identified using the mean vector $\vec{\eta}$ and direction $\vec{v}$ (both highlighted in Figure 3.29–a). In this way, each cluster is infinitely spread in the color space. This infinite domain of color vectors still in an acceptable distance to the cluster, sometimes adds irrelevant color vectors to the results of query region extraction. Figure 3.28 shows a sample result of LPRE–based query region extraction containing the before–mentioned erroneous activity. Note the existence of the reddish–green points in the extracted region (see Figure 3.28–b) while the query region is totaly free of them. This fact is more visible in the color cube (see Figure 3.28–c). Note that the results of region extraction contains members completely outside the locus of the query region color vectors.

(a) (b) (c)

Figure 3.28: Sample erroneous result of LPRE–based region extraction. (a) Original image and the query region. (b) Extracted region. (c) Color vectors of the query region and the extracted region.

Here, we propose a new distance function which is similar to LPRE, but incorporates limitations to avoid its inefficiency. The new distance is called the *limited linear partial reconstruction error* (LLPRE).

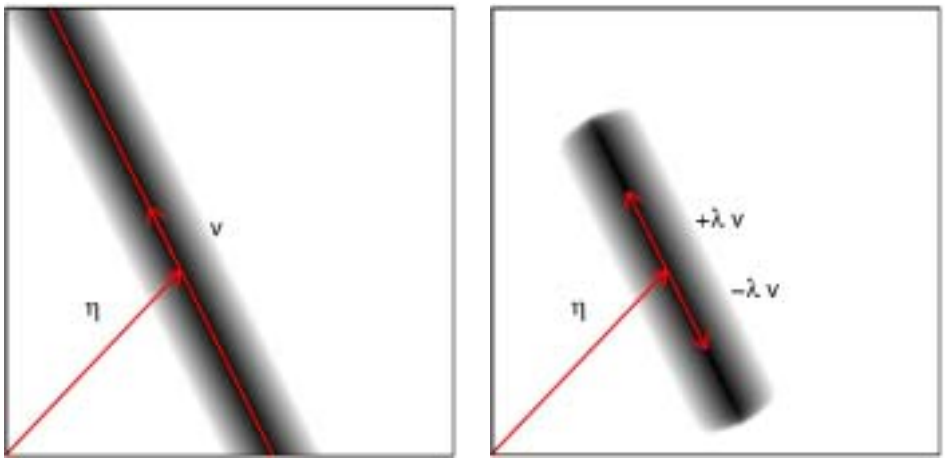
We propose LLPRE distance of the color vector $\vec{c}$ to the set of color vectors r as,

$$\varphi_r(\vec{c}) = \left\| \vec{v}^T(\vec{c}-\vec{\eta})\vec{v} - (\vec{c}-\vec{\eta}) \right\| + f\left(\frac{\vec{v}^T(\vec{c}-\vec{\eta})}{\lambda} \right), \tag{3.4.146}$$

where $f(x)$ is defined as,

$$f(x) = \beta \frac{\alpha^{x^{2n}} - 1}{\alpha - 1}, \tag{3.4.147}$$

where $\alpha > 1$ and β and n are positive. Here, $\vec{\eta}$ and $\vec{v}$ are the expectation vector and the direction of the first principal component of r. Also, $f(x)$ is a weighting function that controls the spread of the cluster. We will talk about the effects of f



(a) (b)

Figure 3.29: Typical distribution of different distances. (a) LPRE. (b) LLPRE.

with more details later. The function $f(x)$ depends on three parameters of n, α, and β. Figure 3.30 shows the typical shape of this function. Function *EHL* computes this function (see Figure 3.31).

Note that while f is an even function it satisfies,

$$f(0) = 0, f(1) = \beta, \tag{3.4.148}$$

$$\lim_{|x|\to\infty} f(x) = 0. \tag{3.4.149}$$

Also, we have,

$$f(2) = \beta\frac{\alpha^{2^{2n}} - 1}{\alpha - 1} = \beta \prod_{k=0}^{2n-1} (\alpha^{2^k} + 1) > 2^{2n}\beta. \tag{3.4.150}$$

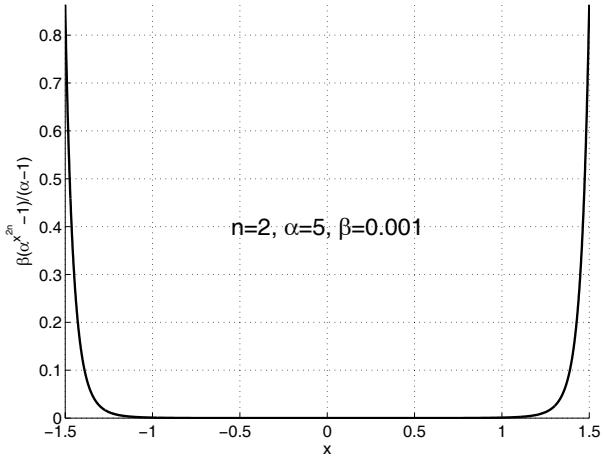


Figure 3.30: Typical shape of the function $f(x)$ defined in (3.4.147) with $n = 2$, $\alpha = 5$, and $\beta = 0.001$.

EHL

```
function y=EHL(x,a,b,n)
%Exponential highlighter computes the values of the function
%b*(a.^(x.^(2*n))-1)/(a-1) at the given x.
%
%Usage:y=EHL(x,a,b,n)
%Inputs: x:Given vector.
%        a,b,n:Parameters (set to default values if left empty).
%Outputs: y:Result.
%Revision 1.01 (c) Arash Abadpour Fall 2004 (Oct. 24 2004)
```

Figure 3.31: Implementation of the *EHL* function.

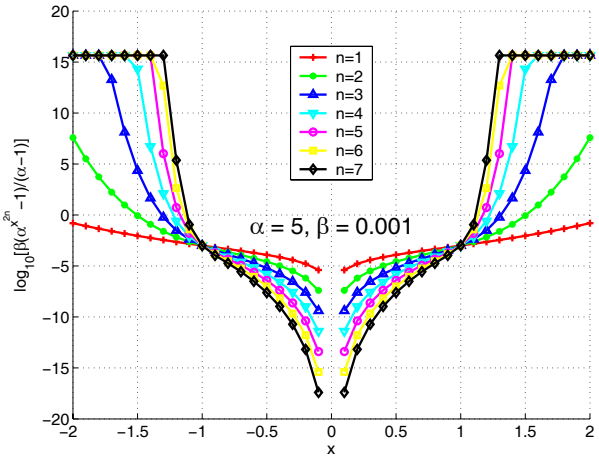


Figure 3.32: Shapes of $f(x)$ defined in (3.4.147) for different values of n with fixed α and β.

Thus, while β controls the value of the function at unity, $f(2)$ depends greatly on n. Figures 3.32, 3.33, and 3.34 illustrate the typical shape of the function for different values of n, α, and β. Note that all curves are drawn in logarithmic scale to help better comparison. In the implementation of the function $f(x)$, a hard threshold is put to avoid it being larger than 10^{15}. As expected from formulation, n controls the growth of the function (see Figure 3.32). Also, α affects the function's growth but not as much as n (see Figure 3.33). Figure 3.34 shows that β is only scaling the function (constant bias in logarithmic scale) and controlling the value of the function in unity.

Lets return to the definition of LLPRE in (3.4.146). Actually, LLPRE adds a second term to the original definition of LPRE. As higher values of a proper distance between individuals and a set show them being less likely members, the second term

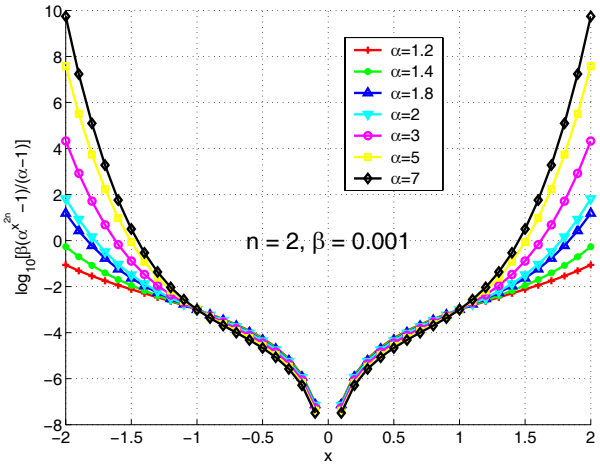


Figure 3.33: Shapes of $f(x)$ defined in (3.4.147) for different values of α with fixed n and β.

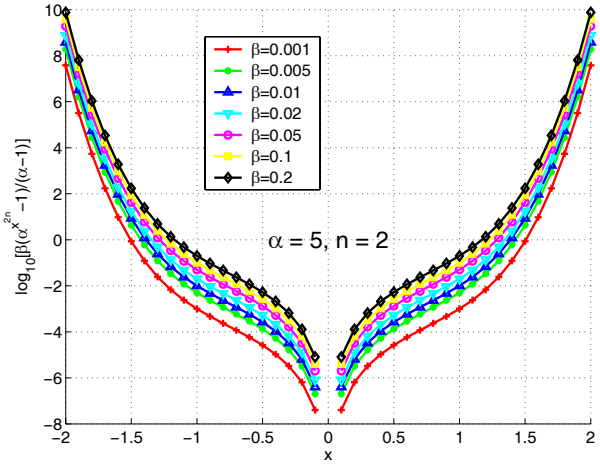


Figure 3.34: Shapes of $f(x)$ defined in (3.4.147) for different values of β with fixed n and α.

in LLPRE is devoted to limit the spread of LPRE. This term contains the result of f on $\vec{v}^T(\vec{c}-\vec{\eta})$ divided by λ. Note that, while LPRE is only concerned with the distance from the given point to the axis of the cluster (see Figure 3.29–a), the $\vec{v}^T(\vec{c}-\vec{\eta})$ term measures the collateral slide of $\vec{c}$ parallel to $\vec{v}$. Due to the tendency of f to give ultra–high values to the points out of the $[-1, 1]$ interval, the second term in LLPRE forces the cluster to get limited in the direction of the main axis. Figure 3.29–b illustrates the results of this new term.

Assume that the image I and the query region r are given and that we are going to perform LLPRE–based query region extraction. In accordance with LPRE criterion for highlighting the good points (those satisfying $\tilde{\tau}_r(\vec{c}) \leq 1$), we propose LLPRE criterion as $\tilde{\varphi}_r(\vec{c}) \leq 1$. Here, the $\tilde{\varphi}_r(\vec{c})$ is defined as,

$$\tilde{\varphi}_r(\vec{c}) = \frac{\varphi_r(\vec{c})}{E_{\vec{x}\in r}\Big\{\tau_r(\vec{c})\Big\}}, \tag{3.4.151}$$

where the cluster spread λ is defined as,

$$\lambda = E_{\vec{x}\in r}\left\{\left|\vec{v}^T(\vec{x}-\vec{\eta})\right|\right\}. \tag{3.4.152}$$

It is experimentally observed that setting the parameters of f as $n = 2$, $\alpha = 5$, and $\beta = 0.001$ gives promising results for all test images.

3.5 Segmentation

Segmentation is an important task in image processing. Hence, this thesis performs a especial focus on it. As such, the following section introduces different segmentation approaches: First, Section 3.5.1 introduces the rough segmentation method of bi–tree decomposition. Then Section 3.5.2 introduces the FPCA–based clustering method which is enhanced in Section 3.5.3 using an agent–based approach. The final contribution of this section is the principal color–based method described in Section 3.5.4.

3.5.1 Bi–tree Decomposition

As discussed in Section 2.2.5, having a proper homogeneity criteria, the image can be decomposed into homogenous regions (here, we use $\|\tau_r^\star\|_{r,p}$). Starting with the entire image area, the tree is produced using the homogeneity criteria defined as:

$$h(r) = \begin{cases} 1, & \|\tau_r\|_{r,p} \leq \varepsilon_1 \\ 0, & otherwise \end{cases}, \tag{3.5.153}$$

where ε_1 is a user-selected parameter, mostly in the range of $[1, 10]$ and $\|\tau_r\|_{r,p}$ is one of the homogeneity criteria discussed in Section 3.3. In an $W \times H$ image, the depth of a $w \times h$ block r is defined as:

$$\varrho_r = \max\left\{ \log_2 \frac{W}{w}, \log_2 \frac{H}{h} \right\}, \tag{3.5.154}$$

where no block is permitted to reach to the depth more than a preselected marginal tree depth value ϱ in the range of $[1, 5]$. Note that ϱ_r computes the maximum number of splitting stages needed to produce r out of the entire image.

During the decomposition stage, all the information is saved as a $22 \times N$ matrix called Λ, where N is the number of blocks and each column of Λ consists of x_1, y_1, x_2, y_2, η_1, η_2, η_3, $V_{ij}, i, j = 1, \cdots, 3$ and some reserved parameters. Here, $[\eta_1, \eta_2, \eta_3]^T$ is the expectation vector of the color information of the region r and V_{ij}s are the elements of the PCA Matrix V corresponding to block r.

Here, we propose a new tree decomposition method called bi–tree decomposition. Assume that the image I is fed into the bi–tree decomposition method. If the block r is not homogenous enough, rather than the deterministic choice of the sub–blocks in the quad–tree decomposition method, here, r is divided either horizontally or vertically (see Figure 3.35), in the way that the total non–homogeneity of the result gets the least possible. In this way if,

$$\|\tau_{r_1}\|_{r_1,p} + \|\tau_{r'_1}\|_{r'_1,p} < \|\tau_{r_2}\|_{r_2,p} + \|\tau_{r'_2}\|_{r'_2,p}, \tag{3.5.155}$$

and the depth limitation permits, the block will be split vertically and otherwise (if the depth limitation is met), it is split horizontally. In the new method, the rectangular clipping is reserved while the block shape changes such that it best fits to the image details.

Any tree decomposition method (containing the quad–tree and the bi–tree), when considered as a black box, processes the input image (in terms of given values of ε_1 and ϱ), resulting in n blocks. No quantitative measure for tree decomposition performance

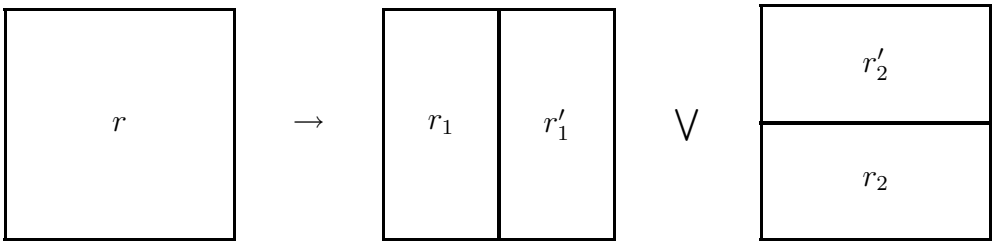


Figure 3.35: Bi–tree decomposition method.

analysis exists in the literature. Here, we propose three measures, concerning the non–homogeneity of the resulting blocks, the average size of them, and the size spread.

It is crucial to measure the performance of a tree decomposition method in terms of giving blocks with the non–homogeneity values not much bigger than the initial value of ε_1. Here, we propose the average homogeneity factor defined as,

$$\tilde{\varepsilon}_1 = \sum_{i=1}^{n} \frac{\|r_i\|}{\|I\|} \|\tau_{r_i}\|_{r_i,p}, \tag{3.5.156}$$

where $\|r\|$ denotes the area of r. Note that $\tilde{\varepsilon}_1$ measures the average homogeneity of the obtained regions weighted by their areas.

Also, it is important to reject methods resulting in many small regions, even if they are all homogenous to a great extent. One step of both horizontal and vertical splitting of R (as performed in quad–tree decomposition method), results in block r satisfying $\|r\| = \frac{1}{4}\|R\|$. Thus, it is reasonable to define,

$$\tilde{\varrho}_r = \log_4 \frac{\|I\|}{\|r\|}, \tag{3.5.157}$$

as the number of splitting steps needed to acquire region r out of image I. Deriving

(3.5.157) gives:

$$\tilde{\varrho}_r = \log_4 \frac{WH}{wh} = \frac{1}{2}(\log_2 \frac{W}{w} + \log_2 \frac{H}{h}). \tag{3.5.158}$$

Comparing (3.5.158) with (3.5.154) shows the similarity between ϱ_r and $\tilde{\varrho}_r$. It is worth to mention that we were forced to use the max operator in (3.5.154) (to avoid ultra thin tall blocks), but we used the mean operator in (3.5.158), to increase the precision of $\tilde{\varrho}_r$; the two 3×10 and 3×20 blocks, in no sense, have the same depth while their corresponding values of ϱ are identical. We propose the expectation of $\tilde{\varrho}_r$ as the average depth of the tree, defined as:

$$\tilde{\varrho} = \frac{1}{n}\sum_{i=1}^{n} \log_4 \frac{\|I\|}{\|r_i\|}. \tag{3.5.159}$$

Assume Deriving (3.5.159) as:

$$4^{-\tilde{\varrho}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\|r_i\|}{\|I\|}}. \tag{3.5.160}$$

Having in mind that the sum of the areas of all sub–blocks equals the area of the original image,

$$\sum_{i=1}^{n} \frac{\|r_i\|}{\|I\|} = 1, \tag{3.5.161}$$

shows that $4^{-\tilde{\varrho}}$ is the geometrical median of n values with summation equal to one. Note that for a fixed value of n, $4^{-\tilde{\varrho}}$ gives its largest value (or similarly $\tilde{\varrho}$ gives its smallest value) for the case of equality:

$$\frac{\|r_i\|}{\|I\|} = c \rightarrow \|r_i\| = \frac{1}{n}\|I\|. \tag{3.5.162}$$

This results in,

$$\tilde{\varrho} = \frac{1}{n} n \log_4 n = \log_4 n. \tag{3.5.163}$$

Now, we define,

$$\tilde{\varphi} = \frac{\tilde{\varrho}}{\log_4 n} = \frac{\sum_{i=1}^{n} \log_4 \frac{\|r_i\|}{\|I\|}}{n \log_4 n}. \tag{3.5.164}$$

Where, $\tilde{\varphi}$ gets to its minimum value (unity) when all blocks have the same size. Also, for values of $\tilde{\varphi} \simeq 1$, we have; $\tilde{\varrho} \simeq \log_4 n$. We propose computing $\tilde{\varphi}$ as the criteria for the proper distribution of the area of tree nodes. Note that a tree decomposition methods is preferred that leads to $\tilde{\varepsilon}_1 \leq \varepsilon_1$ and $\tilde{\varrho} \leq \varrho$ with values of $\tilde{\varphi}$ near unity.

There is a neat trick to enhance the performance of the tree decomposition methods which are based on the LPRE criterion. Here, the homogeneity criterion is defined as,

$$\delta = \frac{1}{n} \sum_{i=1}^{n} \|\vec{x}_i - \vec{\eta} - \vec{v}^T(\vec{x}_i - \vec{\eta})\vec{v}\|^2 = \frac{1}{n} \left\{ \sum_{i=1}^{n} \|\vec{x}_i - \vec{\eta}\|^2 - \sum_{i=1}^{n} \left[\vec{v}^T(\vec{x}_i - \vec{\eta})\right]^2 \right\} \tag{3.5.165}$$

or

$$\delta = \frac{1}{n} \sum_{i=1}^{n} \|x_i\|^2 - \|\vec{\eta}\|^2 - \frac{1}{n} \sum_{i=1}^{n} \left[v^T \vec{x}_i\right] + \left[\vec{v}^T \vec{\eta}\right]^2. \tag{3.5.166}$$

Note that in MATLAB notations we have,

$$\sum_{i=1}^{n} \left[v^T \vec{x}_i\right] = sum(vv^T .* \sum_{i=1}^{n} \vec{x}_i \vec{x}_i^T). \tag{3.5.167}$$

Also note that $\sum_{i=1}^{n} \vec{x}_i^T \vec{x}_i$ equals the sum of diagonal terms of $\sum_{i=1}^{n} \vec{x}_i \vec{x}_i^T$. Hence, for performing the tree decomposition to depth ϱ it suffices to cut the entire image into

DoPCASeg

```
function [R,Change]=DoPCASeg(I,r,depth,method,percent)
%Do Principle Component Analysis is an internal function that is called recursively
%to split the entire image into regions based on PCA reconstruction error
%
%Usage:     [R,Change]=DoPCASeg(I,r,dy,dx,method,percent)
%Inputs:    I: Original color image Image of type uint8 with the size of
%             h*w*3
%          r: Region information
%          depth:Minimum permisable depth.
%          Method:2:Bitree decomposition
%                 4:Quadtree decomposition
%          percent:Inclusion percent multiplied by 100. (-1 forces to use mean)
%Outputs:   R: Region information
%          Change: R==r?
%
%Revision 4.2 (c) Arash Abadpour Spring 2004 (1383/2/22)
```

Figure 3.36: Implementation of the *DoPCASeg* function.

$2^{2\varrho}$ squares of size $\frac{H}{2^{-\varrho}} \times \frac{W}{2^{-\varrho}}$ and then to compute and store $E\{x\}$ and $E\{xx^T\}$ in them.

The set of functions *DoPCASeg*, *DoPCASeg12*, *DoPCASeg2*, *DoPCASeg4*, *PCAMerge*, *PCASegment*, *PCS_CONSTANTS*, *PUTR*, *SetR*, *setri*, *sets*, *merge*, *merged*, and *depthof* are designed to perform tree decomposition on color images.

3.5.2 FPCA–based Clustering (FPCAC)

When attempting to segment an image, the first step is to find a few color–classes that define the image content in an intuitive way. As stated in Section 3.3, the LPRE

DoPCASeg12

```
function [R,Change]=DoPCASeg12(I,r,depth,percent)
%Do Principle Component Anaysis is an internal function that is called recursively
% to split the entire image into regions based on PCA reconstruction error
%
%Usage:      [R,Change]=DoPCASeg12(I,r,dy,dx,percent);
%Inputs:     I: Original color image Image of type uint8 with the size of
%              h*w*3
%           r: Region information
%           depth:Minimum permisable depth.
%                 4:Quadtree decomposition
%           percent:Inclusion percent multiplied by 100. (-1 forces to use mean)
%Outputs:    R: Region information
%           Change: R==r?
%
%Revision 4.3 (c) Arash Abadpour Spring 2004 (1383/2/22)
```

Figure 3.37: Implementation of the *DoPCASeg12* function.

DoPCASeg2

```
function [R,Change]=DoPCASeg2(I,r,depth,percent)
%Do Principle Component Anaysis is an internal function that is called recursively
%to split the entire image into regions based on PCA reconstruction error
%
%Usage:      [R,Change]=DoPCASeg2(I,r,dy,dx,percent)
%Inputs:     I: Original color image Image of type uint8 with the size of
%              h*w*3
%           r: Region information
%           depth:Minimum permisable depth.
%                 4:Quadtree decomposition
%           percent:Inclusion percent multiplied by 100. (-1 forces to use mean)
%Outputs:    R: Region information
%           Change: R==r?
%
%Revision 4.51 (c) Arash Abadpour Spring 2004 (1383/2/23)
```

Figure 3.38: Implementation of the *DoPCASeg2* function.

DoPCASeg4

```
function [R,Change]=DoPCASeg4(I,r,depth,percent)
%Do Principle Component Anaysis is an internal function that is called recursively
%to split the entire image into regions based on PCA reconstruction error
%
%Usage:      [R,Change]=DoPCASeg(I,r,dy,dx,percent);
%Inputs:     I: Original color image Image of type uint8 with the size of
%              h*w*3
%           r: Region information
%           depth:Minimum permisable depth.
%           percent:Inclusion percent multiplied by 100. (-1 forces to use mean)
%Outputs:    R: Region information
%           Change: R==r?
%
%Revision 3.5 (c) Arash Abadpour Spring 2004 (1383/2/22)
```

Figure 3.39: Implementation of the *DoPCASeg4* function.

PCAMerge

```
function [R,t,X,S]=PCAMerge(r,er,I,simple,percent)
%Function PCA merge compses the results of a decomposition process.
%Usage:[R,t,X,S]=PCAMerge(r,er,I,simple,percent)
%Inputs:    r:Decomposed regions.
%          er:Margin
%          I:Original image
%          simple:Simple method for merging
%          percent: Inclusion percent multiplied by 100. (-1 forces to use mean)
%Outputs:   R:Resulting regions
%          t:Elapsed time
%          X:Merged regions
%          S:RAG in an sparse representation
%Revision 1.22 Arash Abadpour Summer 2004 (1383/5/14)
```

Figure 3.40: Implementation of the *PCAMerge* function.

measurement distance defined as,

$$\Psi(\vec{c},[\vec{\eta},\vec{v}]) = \|(\vec{c}-\vec{\eta}) - \vec{v}'(\vec{c}-\vec{\eta})\vec{v}\|^2, \tag{3.5.168}$$

results in a good subjective clustering of color vectors. In this approach the cluster model is a cylinder with its central axis having $\vec{\eta}$ and parallel with $\vec{v}$ (see Figure 3.49). Also, the dual function $\Upsilon(\tilde{X})$ computes the fuzzy expectation and the first fuzzy principal component of $\tilde{X}$ as new values of $\vec{\eta}$ and $\vec{v}$, (as discussed in Section 3.1.1). It must be emphasized that this formulation results in an special case of FCV ($r = 1$) [236]. Here, we propose to assign a threshold, ε, on the changes of the coordinates of cluster centers as the halting condition, (as used generally in the fuzzy clustering

PCASegment

```
function [r,t,Er,Sz,er,sz]=PCAsegment(I,Depth,er,method,percent,reject);
%Function PCA segment decomposes the givenimage with one of decomposition
%methods.
%Usage: [r,t,Er,Sz,er,sz]=PCAsegment(I,Depth,er,method,percent,reject);
%Inputs:        I:Original image
%              Depth:Tree depth, -1 will be treated as automatic
%              er:Margin percentile.
%              Method:2:Bi-tree, 4:Quad-tree, 12:Bi2-Tree
%              percent: Inclusion percent multiplied by 100. (-1 forces to use mean)
%              reject: Reject the final regions not as homogenous as
%              should be?
%Outputs:       r:Resulting regions
%              t:elapsed time
%              Er:error values.
%              Sz:Sizes.
%              er:Mean error.
%              sz:Mean size;
%Revision 5.03 Arash Abadpour Summer 2004 (1383/5/14)
```

Figure 3.41: Implementation of the *PCASegment* function.

PUTR

```
function R=PUTR(r,n,r1)
%Put Region removes n-th object from r and put r1 in it's place
%
%Usage:R=PUTR(r,n,r1)
%Inputs:     r: Region information
%           n: Index to the region to be deleted
%           r1: New regions to be added
%Outputs:    R: Region information
%Revision 1.0 (c) Arash Abadpour 1382
```

Figure 3.42: Implementation of the *PUTR* function.

SetR

```
function  R=SetR(i,Ic,E,v1,v2,v3,r)
%Set Region sets a set of values in the sparse matrix complying the format
%of the sparse matrix
%
%Usage:R=SetR(i,Ic,E,v1,v2,v3,r);
%Inputs:     i: Index to the region.
%           Ic: List of the regions merged with i.
%           E,v1,v2,v3: PCA components.
%           r:Region information
%Outputs:    R: Region information
%PCA stands for Principle Component Analysis
%See PCA and PCAMASK for more information
%Revision 1.1 (c) Arash Abadpour Summer 2004 (1383/5/14)
```

Figure 3.43: Implementation of the *SetR* function.

setri

```
function R=setri(i,E,v1,v2,v3,r)
%Set Region Index sets a single value in the sparse matrix complying
%the format of the sparse matrix
%
%Usage:R=setri(i,E,v1,v2,v3,r)
%Inputs:     i: Index to the region.
%           E,v1,v2,v3: PCA components.
%           r:Region information
%Outputs:    R: Region information
%PCA stands for Principle Component Analysis
%See PCA and PCAMASK for more information
%Revision 1.1 (c) Arash Abadpour Summer 2004 (1383/5/14)
```

Figure 3.44: Implementation of the *setri* function.

sets

```
function S=sets(s,i,j,d)
%Set Sparse puts a value in the sparse matrix complying the format
%of the sparse matrix
%
%Usage:S=Sets(s,i,j,d)
%Inputs:     s: Sparse matrix.
%           i,j: Indexes.
%           d: New value.
%Outputs:    S: Sparse matrix.
%Revision 1.0 (c) Arash Abadpour 1382
```

Figure 3.45: Implementation of the *sets* function.

merge

```
function S=merge(s,i,Ic,j,Jc);
%Merge merges al regions of s encountered in Ic and Jc and also the
%i and j all together
%
%Usage:S=merge(s,i,Ic,j,Jc)
%Inputs:     s: Sparse matrix describing the double-graph connection.
%           i,j: Index of desired regions
%           Ic,Jc: Index of regions previously merged with i and j
%           respectively
%Outputs:    S: Sparse matrix describing the double-graph connection.
%
%Revision 1.0 (c) Arash Abadpour 1382
```

Figure 3.46: Implementation of the *merge* function.

merged

```
function I=merged(S,i)
%Merged gives all regions that are merged t the i-th region according to
%the data available in the sparse matrix S.
%
%Usage:I=merged(S,i);
%Inputs:     S: Sparse matrix describing the double-graph connection.
%           i: Index of the desired region
%Outputs:    I:List of regions previously merged to the i-th region
%
%Revision 1.0 (c) Arash Abadpour 1382
```

Figure 3.47: Implementation of the *merged* function.

depthof

```
function r=depthof(x,X)
%Function depth of computes the number of slicings required to reach to x,
%when operating on X
%Usage: r=depthof(x,X);
%Inputs:     x:Size.
%            X:Original size.
%Outputs:    r:Depth.
%Revision 1.0 Arash Abadpour Spring 2004 (1383/2/22)
```

Figure 3.48: Implementation of the *depthof* function.

theory [334]).

We face the segmentation problem as applying maximum likelihood on fuzzy membership maps to relabel the image pixels, in the way that it results in meaningful geometrical continuous segments. Consider, clustering an image I, into c clusters. The result is the set of c images $J_i, i = 1 \cdots c$ which show the likelihood of each pixel to each of the c clusters. Note that J_i satisfies that,

$$\forall x, y : \sum_{i=1}^{n} J_i(x, y) = 1. \tag{3.5.169}$$

Now, consider a $p \times p$ smoothing convolution kernel M. Note that M satisfies,

$$\sum_{i=1}^{p} \sum_{i=1}^{p} M_{ij} = 1. \tag{3.5.170}$$

Hence, Applying M to each J_i independently to acquire $\tilde{J}_i$, the new membership maps will also satisfy the normality condition. Thus, $\tilde{J}_i, i = 1 \cdots c$ can be considerd

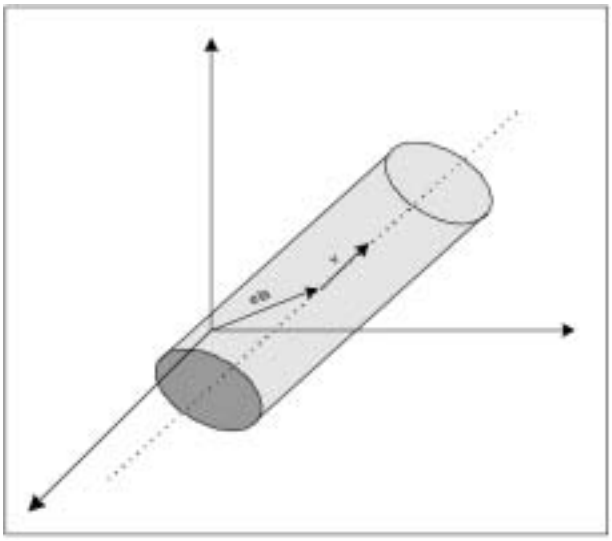


Figure 3.49: Proposed cylindrical clusters.

as the smoothed likelihood to the c clusters. The main benefit of using $\tilde{J}_i$ over J_i is the smoother resulting segments. The crisp segmentation results are obtained using the ML. Here, we propose to use an averaging kernel as M. Different versions of this algorithm are developed in the files *FPCAC3Dp*, *FPCAC3dp*, *FFPCAC3D*, *FPCAC3dre*, and *LFPCAC3Dp*.

3.5.3 Agent–based FPCAC

Here, an agent–based approach for FPCA–based color image segmentation is introduced. Using sets of cooperative agents and their competition, a novel color image segmentation is proposed.

FPCAC3d

```
function [X,Y,Z,V,J,t,loop,K,OJ,Kc]=
                FPCAC3D(I,thr,C,m,vis,map,k,step,compute_OJ)
%Function FPCAC3D perform clustering of color vectors.
%Usage:     [X,Y,Z,V,J,t,loop,K,OJ,Kc]=
%                       FPCAC3D(I,thr,C,m,vis,map,k,step,compute_OJ)
%Inputs:    I:Original image.
%          thr:Threshold.
%          C:Cluster count.
%          m:Fuzziness.
%          vis:visualize?
%          map:Do MAP?
%          k:Convloluion kernel size (0:igonre!).
%          step: Sampling Step.
%          compute_OJ:Compute objective function value?
%Outputs:   X,Y,Z:Cluster center.
%          V:Cluster axis.
%          J:MAP.
%          t:elapsed time.
%          loop:Iterations.
%          K:Filtered membership values.
%          OJ:The objective function value.
%          Kc:Crisp Filtered Memberhsip values.
%!!!It is strongly  recomended to use the function FPCAC3Dp using alpha=0!!!
%Revision 1.33 Arash Abadpour Fall 2004 (October 23, 2004)
```

Figure 3.50: Implementation of the *FPCAC3d* function.

FPCAC3Dp

```
function [X,Y,Z,V,J,t,loop,K,OJ,Kc,Xi,Yi,Zi,Vi,VV]=
            FPCAC3Dp(I,thr,C,m,vis,map,CK,alpha,step,compute_OJ,Weight)
%Function FPCAC3D perform clustering of color vectors.
%Usage:     [X,Y,Z,V,J,t,loop,K,OJ,Kc,Xi,Yi,Zi,Vi,VV]=
%               FPCAC3Dp(I,thr,C,m,vis,map,CK,alpha,step,compute_OJ,Weight)
%Inputs:    I:Original image.
%          thr:Threshold.
%          C:Cluster count.
%          m:Fuzziness.
%          vis:visualize?
%          map:Do MAP?
%          CK:Convloluion kernel size (0:igonre!).
%          alpha:Cluster validity strentgh (default :0).
%          step: Sampling Step.
%          compute_OJ:Compute objective function value?
%          Weight:The convolution kernel to perform CFC.
%Outputs:   X,Y,Z:Cluster center.
%          V:Cluster axis.
%          J:MAP.
%          t:elapsed time.
%          loop:Iterations.
%          K:Filtered membership values.
%          OJ:The objective function value.
%          Kc:Crisp Filtered Memberhsip values.
%          Xi,Yi,Zi,Vi:Initialization.
%          VV:FPCA matrices.
%Revision 2.0 Arash Abadpour Spring 2005 (May 6, 2005)
```

Figure 3.51: Implementation of the *FPCAC3Dp* function.

FFPCAC3Dp

```
function [X,Y,Z,V,J,t,loop,K,OJ,Kc,Xi,Yi,Zi,Vi,N]=
        FFPCAC3Dp(I,thr,C,m,vis,map,CK,alpha,compute_OJ,bin,e,m2,singleloop)
%Function FFPCAC3D perform F2C of color vectors.
%Usage:     [X,Y,Z,V,J,t,loop,K,OJ,Kc,Xi,Yi,Zi,Vi,N]=
%     FPCAC3Dp(I,thr,C,m,vis,map,CK,alpha,step,compute_OJ,bin,e,m2,singleloop)
%Inputs:    I:Original image.
%           thr:Threshold.
%           C:Cluster count.
%           m:Fuzziness.
%           vis:visualize?
%           map:Do MAP?
%           CK:Convloluion kernel size (0:igonre!).
%           alpha:Cluster validity strentgh.
%           compute_OJ:Compute objective function value?
%           bin: Number of bins.
%           e:Perform elimination of empty bins.
%           m2:Second fuzziness.
%           singleloop:Just one loop.
%Outputs:   X,Y,Z:Cluster center.
%           V:Cluster axis.
%           J:MAP.
%           t:elapsed time.
%           loop:Iterations.
%           K:Filtered membership values.
%           OJ:The objective function value.
%           Kc:Crisp Filtered Memberhsip values.
%           Xi,Yi,Zi,Vi:Initialization.
%           N:Number of histogram entries.
%Revision 1.21 Arash Abadpour Spring 2005 (May 7, 2005)
```

Figure 3.52: Implementation of the *FFPCAC3Dp* function.

FPCAC3dre

```
function [X2,Y2,Z2,V2,K2,OJ]=
    FPCAC3Dre(X1,Y1,Z1,V1,I2,thr,C,m,vis,map,kernel,K1,compute_OJ)
%Function FPCAC3D re tunes the clusters.
%Usage:      [X2,Y2,Z2,V2,K2]=
%    FPCAC3Dre(X1,Y1,Z1,V1,I2,thr,C,m,vis,map,kernel,K1)
%Inputs:  X2,Y2,Z2:Cluster center.
%           V2:Cluster axis.
%           I:Original image.
%           thr:Threshold.
%           C:Cluster count.
%           m:Fuzziness.
%           vis:visualize?
%           map:Do MAP?
%           k:Convloluion kernel size (0:igonre!).
%           K1:Membership map.
%           compute_OJ:Compute objective function value?
%Outputs:   X2,Y2,Z2:Cluster center.
%           V2:Cluster axis.
%           J2:MAP.
%           K2:Filtered membership values.
%           OJ:The objective function value.
%Revision 1.1 Arash Abadpour Fall 2004 (Oct. 23,2004)
```

Figure 3.53: Implementation of the *FPCAC3dre* function.

LFPCAC3Dp

```
function [X,Y,Z,V,J,t,loop,K,OJ,Kc,lambda]=
  LFPCAC3Dp(I,thr,C,m,vis,map,k,alpha,step,compute_OJ,Xi,Yi,Zi,Vi)
%Function LFPCAC3D perform clustering of color vectors using the
%limited distance.
%Usage:      [X,Y,Z,V,J,t,loop,K,OJ,Kc,lambda]=
%       LFPCAC3Dp(I,thr,C,m,vis,map,k,alpha,step,compute_OJ,Xi,Yi,Zi,Vi)
%Inputs:    I:Original image.
%           thr:Threshold.
%           C:Cluster count.
%           m:Fuzziness.
%           vis:visualize?
%           map:Do MAP?
%           k:Convloluion kernel size (0:igonre!).
%           alpha:Cluster validity strentgh.
%           step: Sampling Step.
%           compute_OJ:Compute objective function value?
%           Xi,Yi,Zi,Vi:Initialization.
%Outputs:   X,Y,Z:Cluster center.
%           V:Cluster axis.
%           J:MAP.
%           t:elapsed time.
%           loop:Iterations.
%           K:Filtered membership values.
%           OJ:The objective function value.
%           Kc:Crisp Filtered Memberhsip values.
%           lambda:Cluster spread.
%Revision 1.0 Arash Abadpour Fall 2004 (October 25, 2004)
```

Figure 3.54: Implementation of the *LFPCAC3Dp* function.

Corporative Agents

As shown in Figure 3.55, the proposed agent contains two characteristic vectors of $\vec{\eta}$ and $\vec{v}$. While $\vec{\eta}$ controls the expectation vector of the agent, $\vec{v}$ shows its direction and it always satisfies $\|\vec{v}\| = 1$. For a given data cloud, $X = \{\vec{x}_i | i = 1, \cdots, n\}$, each agent searches for a cylindrical structure and fits itself into it. In this manner, each agent goes towards minimizing the objective function defined as,

$$J_a = \sum_{i=1}^{n} p_i^m \Psi(a, \vec{x}_i), \tag{3.5.171}$$

where m is the fuzziness factor, p_i denotes the fuzzy likelihood of $\vec{x}_i$ to the agents vicinity, and $\Psi(a, \vec{x}_i)$ shows the distance from the agent to the vector $\vec{x}_i$. The agent a sees the color vector $\vec{x}$, at the distance computed as,

$$\Psi(a, \vec{x}) = \|\vec{x} - \vec{\eta}_a - \vec{v}_a^T(\vec{x} - \vec{\eta}_a)\vec{v}_a\|^2. \tag{3.5.172}$$

Note that $\Psi(a, \vec{x})$ measures the residue of $\vec{x}$, when it is estimated by its project on the 1–D space described as,

$$\Omega_a = \{\vec{\eta}_a + \lambda \vec{v}_a | \lambda \in R\}. \tag{3.5.173}$$

It is proved that the best choice for the agent, $a_i = [\vec{\eta}, \vec{v}]$, is the new point $[\vec{\eta}^\star, \vec{v}^\star]$, where $\vec{\eta}^\star$ and $\vec{v}^\star$ are the fuzzy expectation and first fuzzy principal direction of the fuzzy set defined as,

$$\tilde{X} = \{(\vec{x}_i; p_i) | \vec{x}_i \in X\}. \tag{3.5.174}$$

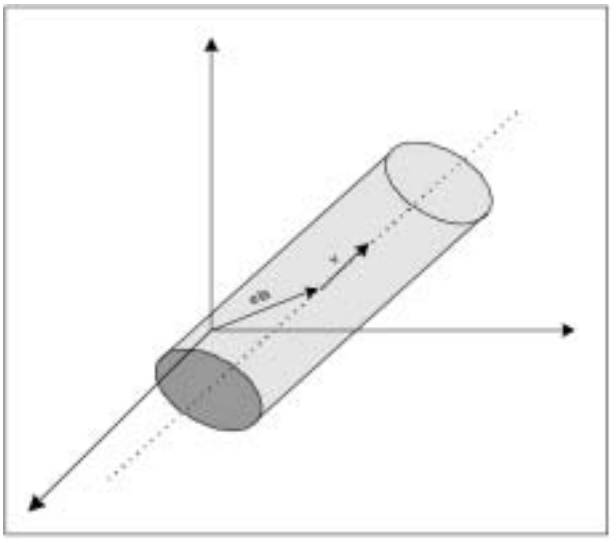


Figure 3.55: Proposed cylindrical agent.

A set of c agents incorporate into a cooperative set when they share their information regarding their distances to the data points. It is proved that, for a set of corporative agents, $A = \{a_1, \cdots, a_c\}$, defining the likelihood of the color vector $\vec{x}$ to the i-th agents cluster as:

$$P_i^A(\vec{x}) = \frac{\Psi(a_i, \vec{x})^{-\frac{1}{m-1}}}{\sum_{k=1}^{c} \Psi(a_k, \vec{x})^{-\frac{1}{m-1}}}, \tag{3.5.175}$$

helps minimizing the objective function J_{a_i} faster. Thus, the objective function for the agent a_i in the cooperative set A is defined as,

$$J_a^A = \sum_{j=1}^{n} P_i^A(\vec{x}_j)^m \Psi(a_i, \vec{x}_j). \tag{3.5.176}$$

A set of cooperative agents is considered to be converged if at an iteration,

$$\max_i \|\vec{\eta}_{a_i}^{\star} - \vec{\eta}_{a_i}\|_\infty < \varepsilon, \tag{3.5.177}$$

for a preselected value of ε.

Competition between Agent Sets

Considering a value of c, a cooperative set of agents, A, is produced, containing c randomly initialized agents and the uniform random value of m in the interval $]1, 2]$. Note that in all stages all agents satisfy the normality of the direction vectors. Here, we propose two ranks for a converged set of cooperative agents. First, the final value of objective function for each agent is a descriptor of the result. Note that the summation of the c objective functions equals,

$$J = \sum_{j=1}^{c} \sum_{i=1}^{n} P_j^A(\vec{x}_i)^m \Psi(a_j, \vec{x}_i). \tag{3.5.178}$$

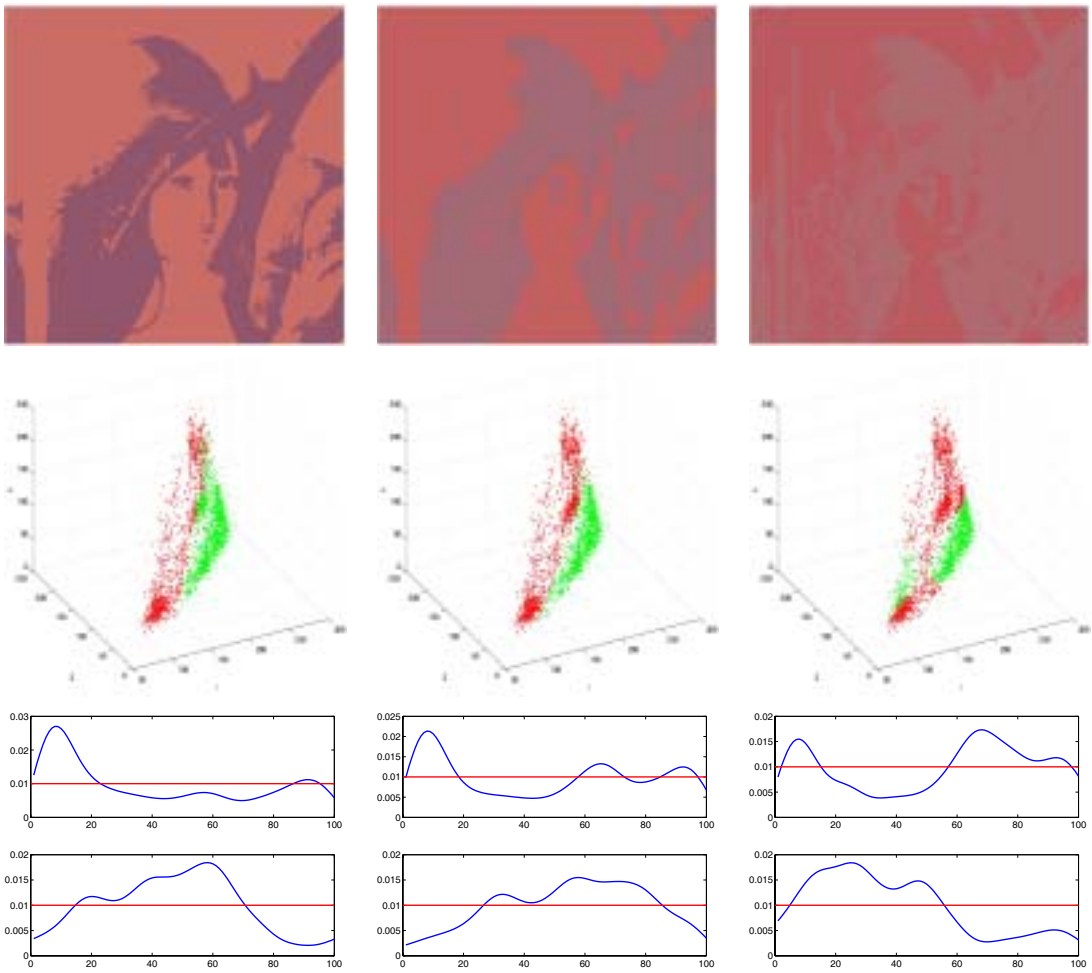
Figure 3.56 shows results of three converged cooperative sets on the image shown in Figure 4.37–a with $c = 2$, $m = 1.07$, and $\varepsilon = 5$. Note the local minima shown in Figure 3.56–c and the corresponding high value of J. Figure 3.57 shows that the low value of J is not a proof for perceptual validity of the clustering results, as among the cases shown in Figure 3.57, Figure 3.57–c has got the lowest J, but it is not preferred to Figure 3.57–a. To solve this ambiguity we propose to use the cluster spread histograms. consider the set of cooperative agents, A, and the data cloud, X. We call $\vec{x}_i \in X$ belongs to the territory of $a_j \in A$, if,

$$\forall k \neq j, P_k^A(\vec{x}_i) < P_j^A(\vec{x}_i). \tag{3.5.179}$$

Assume showing the territory of a_i by X_{a_i}. The projection set of a is defined as,

$$h_a = \{\vec{v}_a^T(\vec{x} - \vec{\eta}_a) | \vec{x} \in X_a\}. \tag{3.5.180}$$

The histogram of h_a shows the spread of the cluster corresponding to the agent a along its axis. To gain noise robustness and to make the histogram of h_a comparable for different agents, we propose to remove the minimum and the maximum vicinity of the data out of h_a. It is empirically observed that estimating the *probability density function*(pdf) with the histogram and removing values of x, satisfying $f(x) \leq 0.02$ or $f(x) \geq 0.98$ is a fast method for increasing the noise robustness of the proposed method. Here, $f(x)$ denotes the estimated pdf. The histogram of the resulting data which is computed in 100 bins shows a highly non–smooth shape. Thus, an averaging kernel of size 5 is applied on the histogram for τ times and the result is normalized to have unity area. Cutting the result with the average line, the number of segments are counted for each cluster. We call the average number of segments in a cooperative set as the integrity denoted by i. Using the two measures of J and i, different cooperative sets can compete. We propose rejecting the sets with high values of i and then selecting the least J in the remainder. Note that using this method in Figure 3.56 the winner is Figure 3.56–a and in Figure 3.57 the winner is Figure 3.57–a, as desired by human observers.

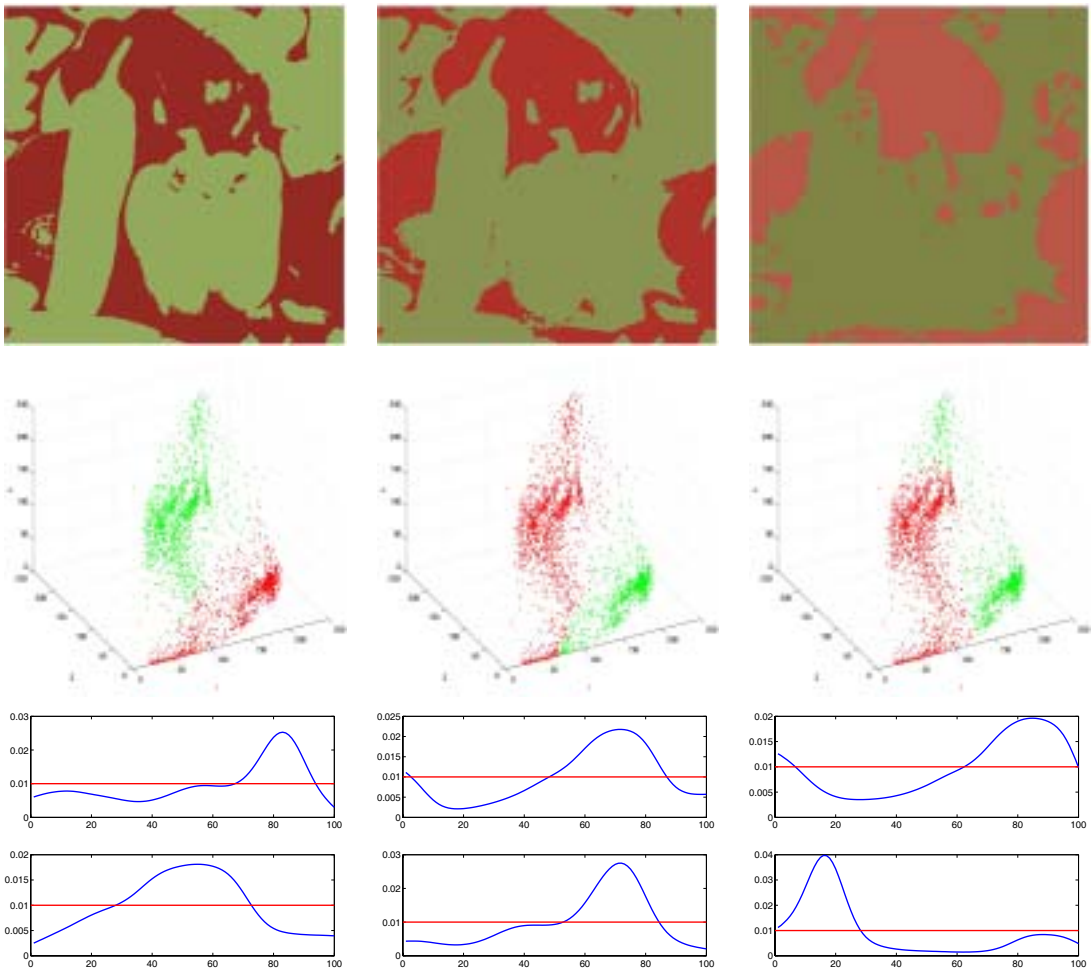


$J = 94.1, i = 1.5$ (a)

$J = 95.8, i = 2$ (b)

$J = 176.8, i = 1.5$ (c)

Figure 3.56: Three converged cooperative sets results for the image shown in Figure 4.37–b with $c = 2$, $m = 1.07$, and $\varepsilon = 5$. Top: The corresponding segments. Middle: The clusters. Bottom:The cluster spread histograms.



$J = 328, i = 1$ (a) $J = 353, i = 1.5$ (b) $J = 290, i = 1.5$ (c)

Figure 3.57: Three converged cooperative sets results for the image shown in Figure 4.37–a with $c = 2$, $m = 1.07$, and $\varepsilon = 5$. Top: The corresponding segments. Middle: The clusters. Bottom:The cluster spread histograms.

Clustering Method

Consider the given color image I in RGB color space and the corresponding 3–D histogram H. Assume selecting the number of the cooperating sets n, and their population, c. We propose to down–sample the given image for the sake of increasing the performance of the agents. For practical reasons, we used a horizontal down–sampling of $20:1$. In each cooperating set, all the agents and the value of m are randomized as described in Section 3.5.3. The best values of ε and τ are empirically proved to be 5 and 10, respectively. Also, it is observed that setting $n = 10c$ is sufficient. Each set works on the image independently and the resulting values of J and i are stored. Among those sets with the smallest value of i, the one with the smallest value of J is considered as the winner. The territories of the agents in the winner set are the clusters. the segmentation result is produced by indexing pixels due to their belonging to the agents territories.

3.5.4 Principal Colors

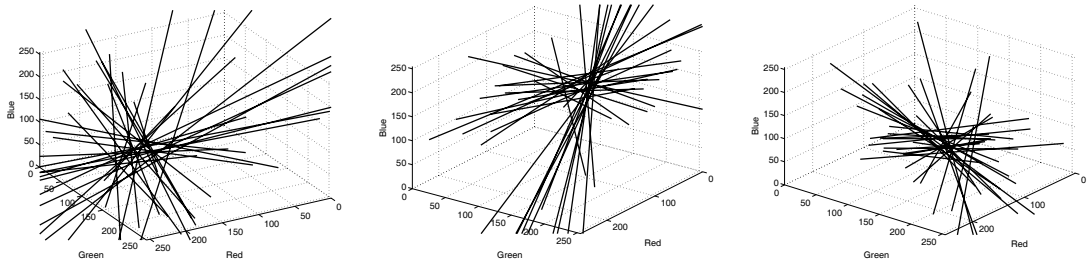
Here, we propose a fairly easy investigation, through which one will have an idea about the existence of the principal colors. We emphasize that this empirical result have to be certified by theoretical investigation. We believe that the increasing number of research projects devoted to processing color images will pave the way to the formal approval of this theory. Figure 3.58 shows three sets of color swatches representing skin, sky, and leaf material. Each set contains 30 images each one a 64×64 JPEG

(a) (b) (c)

Figure 3.58: Freely selected samples of three different materials. (a) Skin. (b) Sky. (c) Leaf.

image with the quality of 100. The skin samples are extracted from erotic images download from free resources in the internet. The sky and leaf samples are captured by a *Cannon A60* digital camera in different times of the day and different places. For each category the lines representing each single swatch are drawn in a common RGB space. Figure 3.59 shows these three sets of lines. Note that in each material, all the lines pass through a small portion of the color space. We call this (virtual) intersection of the lines, the principal color of that material. Figure 3.60 shows the principal color of skin, sky, and leaf extracted by the proposed method. As such, the principal color of a set of realizations of the same material is the closest point to their representing lines. This section incorporates principal color theory with the general fuzzy clustering algorithm described in Section 3.1.3 to propose a new color image segmentation method.

Assume that the $H \times W$ color image I is given. Also, assume that integer c is



(a) (b) (c)

Figure 3.59: The linear representation of the samples shown in Figure 3.58. (a) Skin. (b) Sky. (c) Leaf.

(a) (b) (c)

Figure 3.60: Principal color of the samples shown in Figure 3.58 extracted by the proposed method. (a) Skin. (b) Sky. (c) Leaf.

given which is the number of homogenous segments, I should be split into. For now, assume that both H and W are divided by 2^n, where n is an integer bigger than 2. It is clear that using zero–padding, the same operation is applicable to any image. We propose to cut I into $\frac{WH}{2^{2n}}$ rectangular non–overlapping blocks of $2^n \times 2^n$ pixels. From the $\frac{WH}{2^{2n}}$ blocks, those which are not homogenous are deleted. For the remaining blocks, the PCA representations are saved as the m lines of $l_1, \cdots, l_m$ all in R^3. Now the problem is to find c proper intersections of these lines. Using the notations of the general fuzzy clustering algorithm (see Section 3.1.3) each realization is a line in R^3 and the cluster model is a 3–D point. As such, the distance function between the line l denoted by $\vec{\eta}$ and $\vec{v}$ and the point $\vec{x}$ is defined as,

$$\Psi(l, \vec{x}) = \|(\vec{x} - \vec{\eta}) - \vec{v}^T(\vec{x} - \vec{\eta})\vec{v}\|^2. \tag{3.5.181}$$

In this framework, the dual function Υ is given by (3.1.105). Here, the fuzzy set is $\tilde{L} = \{(l_i; p_i) | i = 1 \cdots, m\}$, where l_i is represented by $\vec{\eta}_i$ and $\vec{v}_i$. The general fuzzy clustering algorithm results in a set of c points in R^3. Each point is the principal color of the respective cluster. note that at the beginning of the process, the non–homogenous blocks were removed. This is done because the principal direction of these blocks are not meaningful. Though, at this point, the process should assign each block to a cluster. Also, note that the results of the general fuzzy clustering algorithm is a set of fuzzy membership values for each single block to each of the clusters. Using a maximum–likelihood operator each block is deterministically assigned to a single

BPCALinSeg.m

```
function [Is,J,K,t,l]=BPCALinSeg(I,C,m,n1,n2,thr,delta,method)
%Function block PCA-based linear segmentation performs a very sophisticated
%thing. Refer to W158 to know what it actually does!
%Usage: [Is,J,K,t,l]=BPCALinSeg(I,C,m,n1,n2,thr,delta,method);
%Inputs:I:Input image.
%       C:Cluster number
%       m:Fuzziness (default 2).
%       n1:Each block used for convergence is 2^n1*2^n1 (default 4).
%       n2:Each block used for segmentation is 2^n2*2^n2 (default 2).
%       thr:Homogeneity Threshold (default 10).
%       delta:Threshold for convergence (Default 0.001).
%       method:Upsampling method ('nearest', ['bilinear']).
%Outputs:Is:Segmented image.
%        J:Fuzzy membership maps.
%        K:Form colors.
%        t:Elapsed time.
%        l:Number of loops.
%Revision 1.01, Arash Abadpour, Spring 2005 (June 6, 2005).
```

Figure 3.61: Implementation of the *BPCALinSeg* function.

cluster. By finding the distance of each omitted block to all clusters, the process assign them with one of the computed clusters. At this point, it only remains to assign clusters to each pixel, and not to each block. As each block is a $2^n \times 2^n$ swatch, performing n times bilinear double–sampling on the resulting index map will yield the desired segmentation. In practice, blocks of size $2^{n_1} \times 2^{n_1}$ are used to find the clusters and then $2^{n_2} \times 2^{n_2}$ blocks are examined to compute the index–map. This process is performed by the function *BPCALinSeg*.

3.6 Compression

Here, two compression procedures are proposed. Section 3.6.1 uses up–sampling, while Section 3.6.2 performs a more professional block–wise interpolation.

3.6.1 Up–Sampling

Based on the proposed image decomposition method (See Section 3.5.1), a new compression method for color images of natural scenes is proposed; As a sub–sampling stage presents in the process, the image is firstly filtered using an ideal low–pass filter to avoid further aliasing:

$$H_1(\zeta_1,\zeta_2)=\begin{cases}1, |\zeta_1|<\frac{1}{2d_x}, |\zeta_2|<\frac{1}{2d_y}\\ 0, otherwise\end{cases} \quad . \tag{3.6.182}$$

Let's call the filtered version of the original image I, as $\tilde{I}$.

One of the decomposition methods discussed in Section 3.5.1 is performed on $\tilde{I}$, resulting in the $13 \times N$ matrix Λ. Now, Λ can be compressed, to be stored in $12N$ bytes, each row contains $\{x_1, y_1, / \log_2 \frac{W}{x_2}/, / \log_2 \frac{Y}{y_2}/\} \cup \{\eta_1, \eta_2, \eta_3, \upsilon_1, \upsilon_2, \upsilon_3\}$, which are all stored as an eight–bit variables, except for x_1, y_1 that are stored as 16–bit variables. Assuming that we are working on a $W \times H$ color image, the size of the file is $3WH$ bytes. We propose computing the gray-scale version of the image (contains WH bytes of data), and replacing the *least significant bit* (LSB) of all bytes with the information in Λ. Now, the user does not perceive the diversified bit. The image now contains $\frac{7}{8}WH + 11N$ bytes of data. If the image is not too scattered the

compression ratio will be about 3. To say more precisely, the compression ratio is less than $\frac{24}{7} \simeq 3.428$, depending on the image complexity.

Now we propose the reconstruction method to compute the uncompressed version of data in color representation. A very simple reconstruction method is using $\vec{\eta}$ and $\vec{v}$ values of any block to colorize its contents (see Section 3.9.1). Here we propose a better method that has shown better quantitative and subjective results. We call the simple method as the zero method while the next method is called the plus method.

Assuming the two $3 \times W \times H$ matrices E and V, and the $W \times H$ matrix C, we set all elements in E, V, and C to zero. Scanning Λ, that we have extracted from the compressed data row by row, we fill the matrices in this way: When processing the block $(x_1, y_1) - (x_2, y_2)$ with the two describing vectors $\vec{\eta} = [\eta_1 \quad \eta_2 \quad \eta_3]^T$ and $\vec{v} = [v_1 \quad v_2 \quad v_3]^T$, for any $x \in [x_1 \dots x_2]$ and $y \in [y_1 \dots y_2]$ that $d_x | x$ and $d_y | y$ and for all $i \in \{1, 2, 3\}$, we set $E[i, \omega_{d_x}(x), \omega_{d_y}(y)] = \eta_i$ and $V[i, \omega_{d_x}(x), \omega_{d_y}(y)] = v_i$. Also we set $C[\omega_{d_x}(x), \omega_{d_y}(y)] = 1$. Where $\omega_{d_x}(x)$ defined as:

$$\omega_\alpha(x) = \alpha \times /\frac{x}{\alpha}/, \tag{3.6.183}$$

is used to align the points on a $d_x \times d_y$ grid. Due to occurrence of round–off, some samples of E and V align the $d_x \times d_y$ grid are lost. Hence, C is used to compensate this information-loss. When setting an element of E, say $E(i, x, y)$, the set defined as:

$$\{(u, v) || u - x| \leq d_x, |v - y| \leq d_y\}, \tag{3.6.184}$$

is searched for zero values of C align the $d_x \times d_y$ grid. If such points are found, the values of $E(i,x,y)$ and $V(i,x,y)$ are copied into appropriate cells of E and V respectively. Now E and V, contain samples of the signals $\tilde{E}$ and $\tilde{V}$ sampled by a cartesian grid $(d_x \times d_y)$. Reconstruction of $\tilde{E}$ and $\tilde{V}$ is easily performed using the ideal low-pass filter defined as:

$$H_2(\zeta_1,\zeta_2) = \begin{cases} d_x d_y, |\zeta_1| < \frac{1}{2d_x}, |\zeta_2| < \frac{1}{2d_y} \\ 0, otherwise \end{cases} \quad . \tag{3.6.185}$$

Using G (the contents of the 7 *most significant bits* (MSB) of the compressed image) as the source image and $\tilde{E}$, $\tilde{V}$ as the color content, by applying either of the two colorizing methods discussed in Section 3.9.1, the color information is reproduced. The proposed compression method is independent of the method that has produced Λ, either be quad–tree decomposition or bi-tree decomposition.

3.6.2 Block–Wise Interpolation

Here, we first propose a fast and reliable method to compute the corresponding $V_{r_{\vec{c}}}$ and $\vec{\eta}_{r_{\vec{c}}}$ for all image pixels. Assume feeding the given image I to the bi–tree (or equivalently the quad–tree) decomposition method. The output of the decomposition method is the matrix Λ containing the coordinates of r_i along with the expectation matrix $\vec{\eta}_i$ and the polarized version of the PCA matrix $(\theta_i, \phi_i, \psi_i)$. Storing this portion of the Λ matrix needs $10n$ bytes. For ordinary values of $n \simeq 200$ in a 512×512 image, Λ will take about $\frac{1}{400}$ of the original image data. Now assume solving the problem

$\{(r_i;\xi_i)|i=1,\cdots,n\}$ using the block–wise interpolation, where ξ_i is the row vector containing η_{i_1}, η_{i_2}, η_{i_3}, θ_i, ϕ_i, and ψ_i. Note that the three values of θ_i, ϕ_i, and ψ_i are of angular type. Assume the solutions of the problem as the functions $\tilde{\eta}_1$, $\tilde{\eta}_2$, $\tilde{\eta}_3$, $\tilde{\theta}$, $\tilde{\phi}$, and $\tilde{\psi}$. Now we compute the functions $\tilde{\vec{\eta}}: R^2 \rightarrow R^3$ and $\tilde{V}: R^2 \rightarrow R^9$, as the value of the expectation vector and the PCA matrix in each pixel, respectively. This leads to the computation of the three eigenimages pc_1, pc_2 and pc_3 are computed (using (3.1.72)). We call the function $\tilde{\vec{\eta}}: R^2 \rightarrow R^3$ as the *expectation map* (Emap) and the polarized version of $\tilde{V}: R^2 \rightarrow R^9$ as the *rotation map* (Rmap), respectively. As the PCA theory states [176], we expect the standard deviation of the three planes to be in descending order. Functions *im2eig* and *eig2im* produce the eigenimages of an image or reconstruct an image from its eigenimages, respectively (see Figures 3.65 and 3.63). These two functions work on the results of the function REmap shown in Figure 3.64. Also, functions Bdecompose.m, *Bim2eig*, *Beig2im*, and *energy* are specifically designed for block PCA–based eigenimage extraction (see Figure 3.65, 3.66, and 3.67).

Consider the image I and its corresponding eigenimages pc_1, pc_2, and pc_3. Due to the energy compaction condition (see (3.1.80)), this scheme is actually an spectral image compression method. Reconstructing the image using two or just one eigenimage(s) gives the compression ratios of $\frac{3}{2}:1$ and $3:1$ (the theoretical margin) respectively. To add the spatial compression ability to the proposed method, we use

im2eig

```
function [e,V,pc]=im2eig(I,E1,E2,E3,THETA,PHI,PSI)
%Function image to eigenimage finds the eigenimages.
%Usage:      [e,V,pc]=im2eig(I,E1,E2,E2,THETA,PHI,PSI)
%Inputs:     I:Original image.
%           E1,E2,E3:Emap.
%           THETA,PHI,PSI:Rmap.
%Outputs:    e:emap.
%           V:Rmap.
%           pc:eigenimages.
%Revision 1.0 Arash Abadpour Spring 1383 (1383/2/27)
```

Figure 3.62: Implementation of the *im2eig* function.

eig2im

```
function L=eig2im(pc,e,V,chno,w,h)
%Function eigenimage to image reconstructs the image neglecting some of the
%eigenimages.
%Usage:      L=eig2im(pc,e,V,chno,w,h)
%Inputs:     pc:Eigenimages.
%           e:Emap.
%           V:Rmap.
%           w,h:Original image sizes.
%Outputs:    L:Reconstructed image.
%Revision 1.0 Arash Abadpour Spring 2004 (1383/2/27)
```

Figure 3.63: Implementation of the *eig2im* function.

REmap

```
function [E1,E2,E3,THETA,PHI,PSI]=REmap(I,r)
%Function Rotation and Expectation map finds the Rmap and Emap
%corresponding to the image I and the decomposition r.
%Usage:      [E1,E2,E3,THETA,PHI,PSI]=REmap(I,r)
%Inputs:     I:Original image.
%           r:Decomposition information.
%Outputs:    E1,E2,E3:Expectation map.
%           THETA,PHI,PSI:Rotation map.
%Revision 1.0 Arash Abadpour Spring 2004 (1383/2/3)
```

Figure 3.64: Implementation of the *REmap* function.

Bim2eig

```
function [e1,e2,e3,t]=Bim2eig(IRd,IGd,IBd,R,G,B,Theta,Phi,Psi)
%Function block image to eigenimage computes the eigenimages.
%
%Usage:    [e1,e2,e3,t]=Bim2eig(IRd,IGd,IBd,R,G,B,Theta,Phi,Psi);
%Inputs:   IRd,IGd,IBd:Image Differences.
%          R,G,B:Upsampled Expectation Maps.
%          Theta,Phi,Psi:Upsamples Rotation Maps.
%Outputs:  e1,e2,e3:Eigenimages.
%          t:Elapsed time.
%Revision 1.01 Arash Abadpour Winter 2005 (Jan. 26, 2004)
```

Figure 3.65: Implementation of the *Bim2eig* function.

Beig2im

```
function [JRd,JGd,JBd,JR,JG,JB,J,t]=Beig2im(e1,e2,e3,R,G,B,Theta,Phi,Psi,SHOW)
%Function block eigenimage to image computes the image.
%
%Usage:
%[JRd,JGd,JBd,JR,JG,JB,J,t]=Beig2im(e1,e2,e3,R,G,B,Theta,Phi,Psi,SHOW);
%Inputs:  e1,e2,e3:Eigenimages.
%         R,G,B:Upsampled Expectation Maps.
%         Theta,Phi,Psi:Upsamples Rotation Maps.
%        SHOW:Visualize the Results.
%Outputs: JRd,JGd,JBd:Image Differences.
%         JR,JG,JB,J:Image.
%         t:Elapsed time.
%Revision 1.0 Arash Abadpour Fall 2004 (Dec. 10, 2004)
```

Figure 3.66: Implementation of the *Beig2im* function.

the PU–PLVQ [205] gray–scale image compression technique for each eigenimage with different compression ratios (see Figure 3.68).

In [205] the authors proposed an efficient (spatial) compression algorithm for gray–level images. In that work, using the *wavelet packets* and the *pyramid lattice vector quantization*, a new compression scheme based on an accurate model for the distribution of the wavelet coefficients is presented (the model is based on the generalized Gaussian distribution). Additionally, a new technique for determining the largest *radius* of the lattice and its *scaling factor*, for both uniform and piecewise–uniform pyramidal lattices, is proposed. The proposed algorithms aimed at achieving the best rate–distortion function by adapting to the statistical characteristics of the source

Bdecompose

```
function [r,g,b,theta,phi,psi,R,G,B,Theta,Phi,Psi,t,IRd,IGd,IBd,sz]=
                                        Bdecompose(I,n,USmethod,Quality)
%Function block decompose decomposes the image I.
%
%Usage:
%[r,g,b,theta,phi,psi,R,G,B,Theta,Phi,Psi,t,IRd,IGd,IBd,sz]=
                                        Bdecompose(I,n,USmethod,Quality);
%Inputs:  I:Original Image.
%         n:Decomposition Level.
%         USmethod:Interpolation method ('bicubic','nearest','bilinear')
%        Quality: Jpeg compression of polarized PCA matrices quality.
%Outputs: r,g,b:Expectation Maps.
%        theta,phi,psi:Rotation Maps.
%        R,G,B:Upsampled Expectation Maps.
%        Theta,Phi,Psi:Upsamples Rotation Maps.
%        t:Elapsed time.
%        IRd,IGd,IBd:Differences.
%        sz:Size of polarized PCA matrices after compression.
%Revision 1.3 Arash Abadpour Winter 2005 (Feb. 23, 2004)
```

Figure 3.67: Implementation of the *Bdecompose* function.

in each subimage. In the lattice parameters optimization algorithm, no assumption about the lattice parameters is made and no training and multi–quantizing is required. Also, in the piecewise–uniform algorithm, the wedge region problem encountered with sharply distributed random sources is resolved. The algorithms adapt to nonstationarity in input images and to the given bit rates. Compared to other available image compression algorithms, these algorithms resulted in higher quality reconstructed images for identical bit rates (for more details see [205]). Here, we propose a novel method for extracting three eigenimages from a single color image. Also, we propose a method to produce the original image out of the eigenimages. Using the spatial compression method introduced in [205], each eigenimage is compressed, while the decompression side performs in the inverse process. The main contribution of this thesis to the method in [205], is adding the spectral compression aspect to the method, while increasing the compression ratio.

As Figure 3.68 shows, the transmitted information contains the compressed versions of the pc_1, pc_2, and pc_3, along with the Λ, α, a, β, and b (for block–wise interpolation). Assuming that the image to be compressed is a $H \times W$ color image, decomposed into n blocks, the total amount of information to be sent equals: $10n$ bytes for storing x_{i1}, x_{i2}, y_{i1}, y_{i2}, η_{i1}, η_{i2}, η_{i3}, θ_i, ϕ_i, and ψ_i plus $WH(\lambda_1^{-1}+\lambda_2^{-1}+\lambda_3^{-1})$ for storing pc_1, pc_2 and pc_3 eigenimages compressed with compression ratios of λ_1, λ_2,

and λ_3, respectively (where $\lambda_1 > \lambda_2 > \lambda_3$). Thus, the total compression ratio equals,

$$\lambda \simeq \frac{3}{\lambda_1^{-1} + \lambda_2^{-1} + \lambda_3^{-1} + \frac{10n}{WH}}. \tag{3.6.186}$$

A nominal value of $\lambda_2 = \lambda_1$ and $\lambda_3 = \infty$ leads to $\lambda \simeq 1.5\lambda_1$. Note that using a pure spatial compression, all three channels must be compressed with almost the same compression ratios, resulting in a total compression ratio of,

$$\tilde{\lambda} = \frac{3WH}{\frac{WH}{\lambda_1} + \frac{WH}{\lambda_1} + \frac{WH}{\lambda_1}} = \lambda_1. \tag{3.6.187}$$

As shown in Figure 3.69, in the decompression process the Emap and the Rmap are computed just like what performed in the encoding process. Using these information along with the decoded versions of pc_1, pc_2, and pc_3, the original image is reconstructed.

3.7 Watermarking

Here, we use the proposed eigenimage framework for proposing two methods for color image watermarking. The first method discussed in 3.7.1 uses the bi–tree decomposition introduced in Section 3.5.1 equipped with the block–wise interpolation proposed in Section 3.1.5. The second one, and the faster one, incorporates FPCA clustering (see Section 3.7.2).

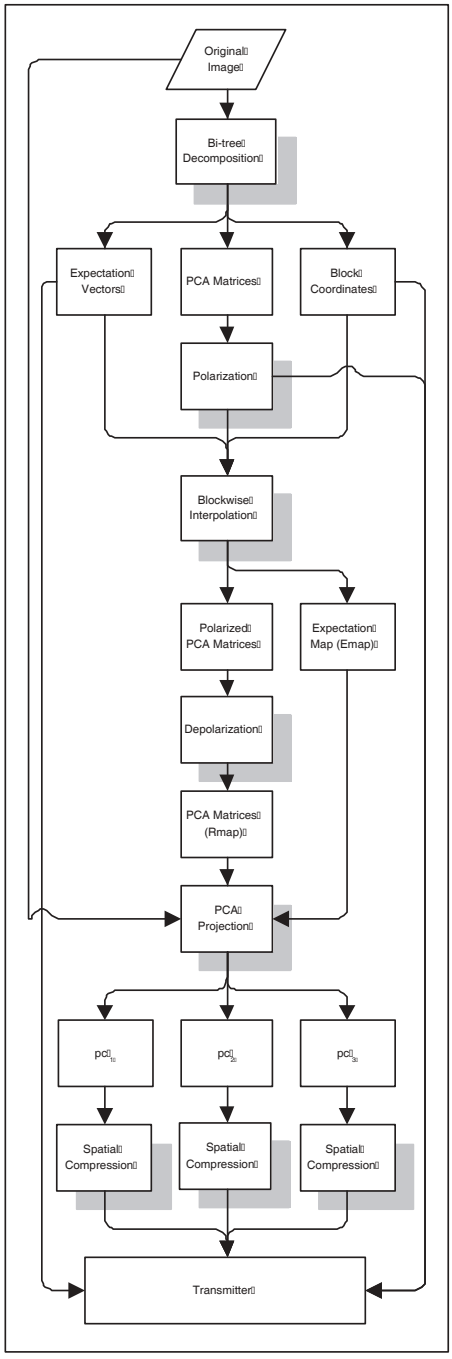


Figure 3.68: Flowchart of the proposed color image compression method.

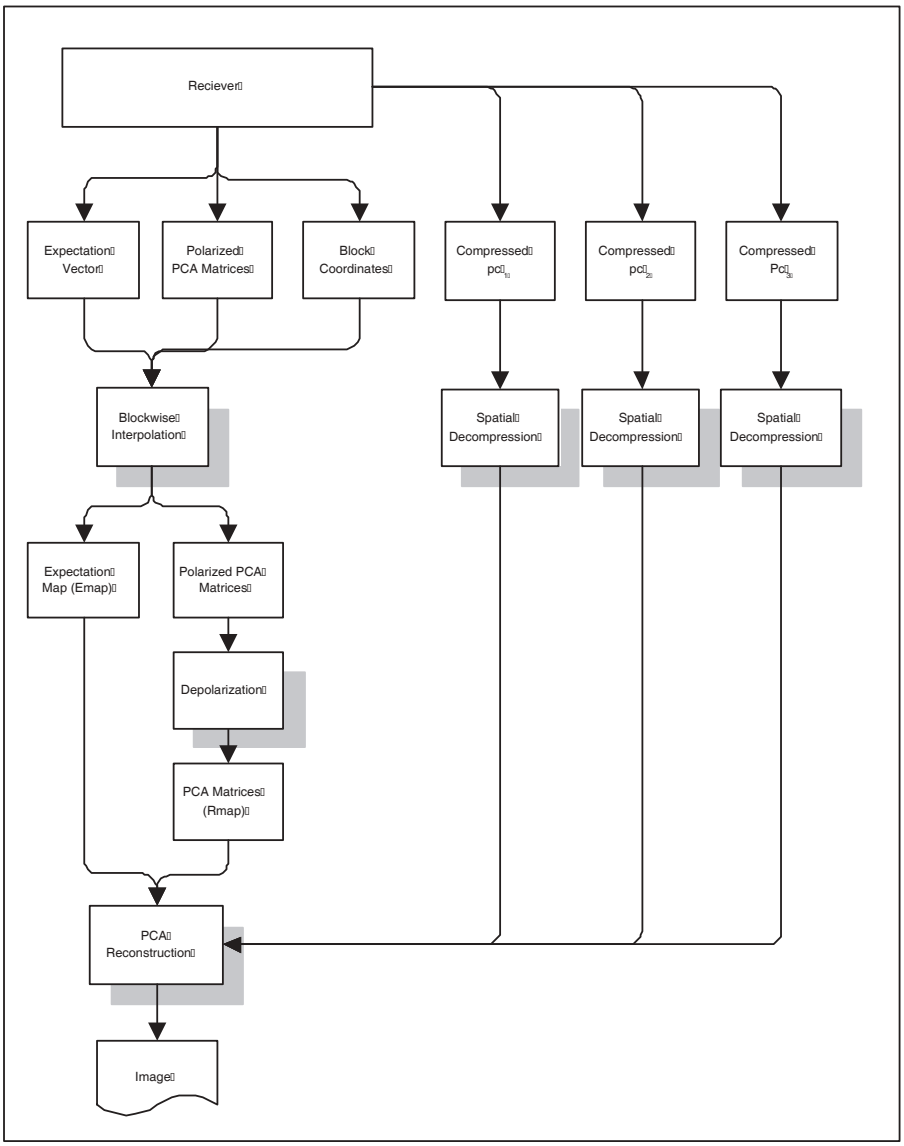


Figure 3.69: Flowchart of the proposed color image decompression method.

3.7.1 Block–Wise Interpolation

Consider the image I with the three eigenimages pc_1, pc_2, and pc_3. Although, there is no orthogonality constraint in the eigenimage theory, but the eigenimage approach can easily be adopted for watermarking purposes. Assume that the gray–scale image W is to be embedded into I as the watermark data. Also, assume that I and W are of the same size $(H \times W)$. First the dynamic range of W is fit into pc_3 domain, as:

$$\tilde{W} = \frac{\sigma_{pc_3}}{\sigma_W}\Big(W - \eta_W\Big). \tag{3.7.188}$$

Replacing the pc_3 with the scaled version of the watermark signal, $\tilde{W}$, the watermarked image, I', is constructed. The process of extracting the watermark is vice versa: compute the eigenimages corresponding to I' as pc'_1, pc'_2, and pc'_3. Now, pc'_3 (when normalized) contains the watermark signal:

$$W' = 255\frac{pc'_3 - (\eta_{pc'_3} - \sigma_{pc'_3})}{2\sigma_{pc'_3}}. \tag{3.7.189}$$

Comparing (3.7.188) and (3.7.189) is helpful. In equation (3.7.188) the watermark signal is scaled to fit the energy range of pc_3. This is crucial to prevent disturbing the balance between the eigenimages in the given image. Equation (3.7.189) performs the inverse transformation by widening the dynamic range of the extracted watermark signal. In this way, the contents of the extracted watermark get recognizable.

3.7.2 Clustering–Based Approach

Assume that the color image I and the grayscale watermark signal W are given. Firstly, we propose to feed I into the FPCAC clustering algorithm proposed in Section 3.5.2. As a result, the pixels of I will be categorized into c classes with the crisp membership maps denoted as $K_i, i = 1, \cdots, c$. Now, produce the sets,

$$\varphi_i = \left\{ \vec{I}_{xy} | K_{ixy} = 1 \right\}, i = 1, \cdots, c, \tag{3.7.190}$$

Where, φ_i contains the color vectors in the i–th cluster and x and y denote the pixel coordinates. As the clusters are cylindrical, the members of each φ_i set can be described using lower dimensional representations (note that $Rank(\varphi_i) = 3$). Denoting the expectation vector of φ_i as $\vec{\eta}_i$ and its PCA matrix as C_i, each member of φ_i can be translated into PCA coordinates as,

$$\vec{c}' = C_i^{-1}(\vec{c} - \vec{\eta}_i), \qquad \vec{c} \in \varphi_i. \tag{3.7.191}$$

To reach to these PCA coordinates for all pixels, each pixel in the image must be processed using the parameters of the cluster to which it belongs. We propose computing,

$$\vec{I}_{1xy} = \sum_{i=1}^{c} K_{ixy} C_i^{-1} (\vec{I}_{xy} - \vec{\eta}_i). \tag{3.7.192}$$

Now I_1 is a three channel image containing the eigenimages of the original image. Denote the three channels of I_1 as i_1, i_2, and i_3.

As shown in Section 3.3 for homogenous sets (like φ_i), neglecting the third principal component or adding proper amount of data to it, and then reconstructing the data back, produces no artifacts and preserves the subjective quality of the image. As such, we propose computing,

$$\vec{I}_{Wxy} = \sum_{i=1}^{c} K_{ixy}(C_i I_{2xy} + \vec{\eta}_i), \tag{3.7.193}$$

where I_2 is a three channel image with its first and second channels being the same as I_1 while its third channel contains the watermark signal or is filled with zeros. To prevent disturbing the visual appeal of the image after watermarking, the watermark signal is scaled to fit the energy range of the third channel. Denoting the third channel as i_3 and the watermark signal as W, this is performed using,

$$\tilde{W} = \frac{\sigma_{i_3}}{2\sigma_W}(W - \eta_W). \tag{3.7.194}$$

Here, η_W and σ_W denote the expectation and standard deviation of the watermark signal (W).

Assume the watermarked image I is given and we are going to extract the (probable) watermark from it. The watermark extraction process contains computing I_1 as (3.7.192) and normalizing the third principal channel (i_3) as,

$$\tilde{i}_3 = 255\frac{i_3 - \eta_{i_3}}{\sigma_{i_3}}. \tag{3.7.195}$$

Most of the times the extracted watermark is visible but further increase of the contrast makes it more appropriate specially when it is needed to print the result or

FPCACim2eig

```
function [Eig,t,J,E,V]=FPCACim2eig(I,C)
%Function FPCAC image to eigen image computes the eigenimages of the given
%image using the FPACA clustering method.
%Usage:      [Eig,t,J,E,V]=FPCACim2eig(I,C)
%Inputs:     I:Original image.
%           C:Cluster count.
%Outputs:    Eig:Eigen images.
%           t:elapsed time.
%           J:Cluster membership mask.
%           E,V:PCA information.
%Revision 1.0 Arash Abadpour Summer 2004 (1383/6/5)
```

Figure 3.70: Implementation of the *FPCACim2eig* function.

show it in low–quality monitors. Functions *FPCACim2eig* and *FPCACeig2im* enable extracting the eigenimages from an image and reconstructing it from eigenimages, respectively (see Figures 3.70 and 3.71).

3.8 Direct Copy Protection

According to the fall of the watermarking methodology, discussed in Section 2.2.12, this section proposes two novel methods for color image copy protection. The first method introduced in Section 3.8.1 includes the first ideas. Section 3.8.2 introduces a more professional approach.

FPCACeig2im

```
function [L,t]=FPCACeig2im(Eig,t,J,E,V,C)
%Function FPCAC image to eigen image computes the eigenimages of the given
%image using the FPACA clustering method.
%Usage:      [I,t]=FPCACeig2im(Eig,t,J,E,V,C)
%Inputs:     Eig:Eigen images.
%           t:elapsed time.
%           J:Cluster membership mask.
%           E,V:PCA information.
%           C:Cluster count.
%Outputs:    I:Original image.
%           t:Elapsed time.
%Revision 1.0 Arash Abadpour Summer 2004 (1383/6/5)
```

Figure 3.71: Implementation of the *FPCACeig2im* function.

3.8.1 Clustering–Based Approach

Figure 3.72 shows the flowchart of the proposed visual encryption/decryption method. Here, for the given image $I_\circ$, the encryption block produces the encrypted demo image $\tilde{I}$. It is done in a manner that $\tilde{I}$ visualizes the contents of $I_\circ$ while containing deliberately embedded distortions that prevent an unauthorized consumer to use it for professional applications. When the authorization is performed, the key K is given to the user. Using K and $\tilde{I}$, the image I is reconstructed so that it perfectly duplicates the original image $I_\circ$. The main contributions of the proposed method are its robust protection against unauthorized duplication and distribution, the low elapsed time for encryption/decryption, high quality of the reconstructed image, and

the low redundancy of K over $I_\circ$.

Assume that the original $H \times W$ image $I_\circ$ is to be encrypted by the proposed method. As shown in Figure 3.72, the FPCA clustering method is performed on $I_\circ$. The results are a set of C color classes $\phi_i = [\vec{\eta}_i, V_i]$ for $i = 1, \cdots, C$ along with C, $H \times W$ binary membership maps $J_1, \cdots, J_C$. For each value of i, the vector $\vec{\eta}_i$ is the fuzzy expectation vector of the color vectors in the i–th cluster (see Section 3.5.2). Also, the V_i is the FPCA matrix of the i–th cluster.

The FPCAC process depends on the fuzziness parameter (m) with default value of 1.05 for color images of the nature (see Section 3.5.2). As here we do not expect a satisfactory classification (in fact, we prefer some levels of degradation), the m value can be freely selected ($m > 1$). We propose to select m randomly using,

$$m = 1 + \Big|N(0,1)\Big|, \tag{3.8.196}$$

where $N(0,1)$ is a zero–mean, unit–variance Gaussian random variable. In fact, m enables the proposed method to give any number of encrypted version of a single image that is needed. Each encrypted image needs its own key to be unleashed.

Here, the main concern is to recolor a given image in an unsupervised manner such that the image content remains understandable, while its quality degrades considerably. Thus, we propose a disturbing multi–swatch recoloring algorithm that is reversible. Assume producing the $H \times W$ index map J with its elements being members of $\{1, \cdots, C\}$. As such, $J_{xy} = i$ iff $J_{ixy} = 1$. As described in Section 3.5.2, there

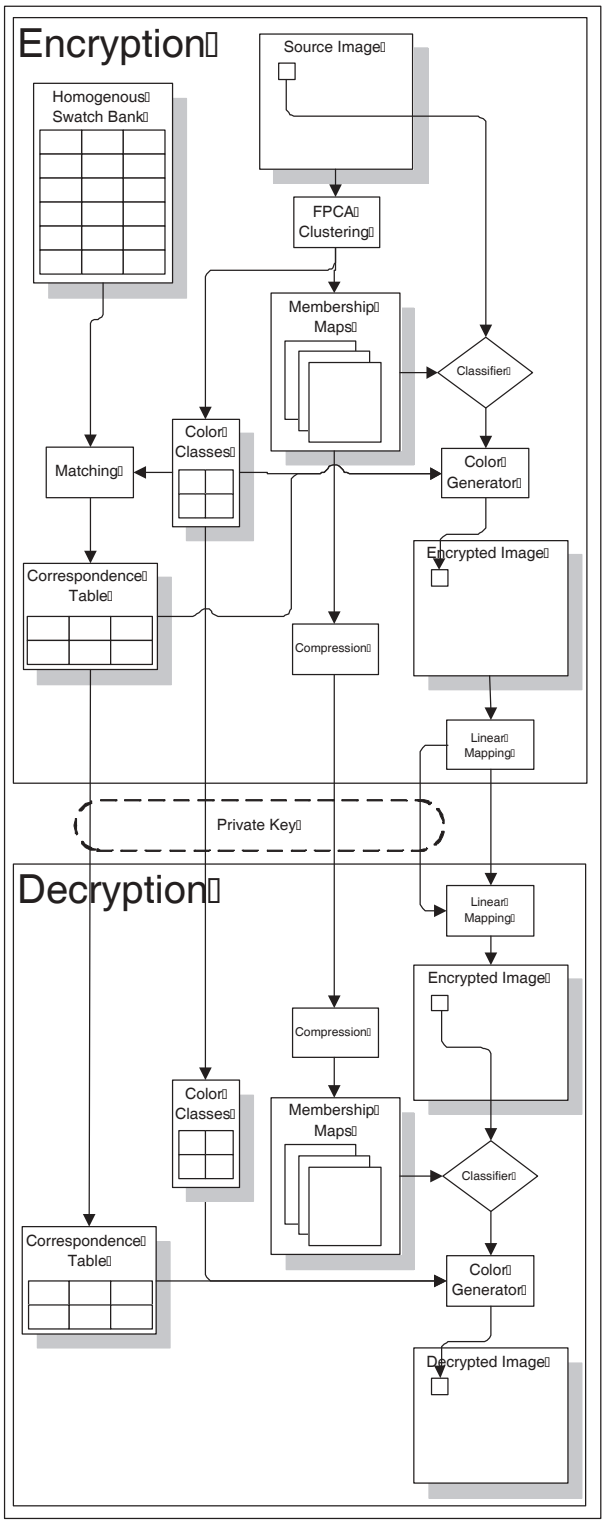


Figure 3.72: Flowchart of the proposed copy protection algorithm.

is one and only one choice for J. Using the single–swatch recoloring method discussed in Section 3.9.2, we construct the encrypted image as,

$$\vec{\tilde{I}}_{xy} = U_i V_i^T (\vec{I}_{\circ xy} - \vec{\eta}_i) + \vec{\rho}_i, i = J_{xy}, \tag{3.8.197}$$

where $[\vec{\rho}_i, U_i]$s for $i = 1, \cdots, C$ are a set of color descriptors and x and y are image coordinates. We will return to the above color descriptors later. For now, it suffices to know that $\vec{\rho}_i$ is a 3×1 vector and U_i is a 3×3 unitary matrix.

In fact, in (3.8.197) each color vector is transferred from the color descriptor to which it mostly belongs to, into a second color descriptor. Having J and $[\vec{\rho}_i, U_i]$s for $i = 1, \cdots, C$, the original image is reconstructed from $\tilde{I}$. Hence, J and $\{[\vec{\rho}_i, U_i] | i = 1, \cdots, C\}$ produce the key (K). The J is a large image and occupies memory, but due to its content, conventional lossless binary compression methods are highly efficient with compressing it. Furthermore, we propose to apply a convolution kernel of radius r on the membership map of the FPCAC, prior to the maximum likelihood process. This is done to produce an index map with more smoothed edges. This will result in an index map with a few holes and less isolated pixels (that declines the performance of the run–length coding).

Also, we propose to compute the redundancy of the key as its volume in compressed format over the volume of the whole image. Thus, the redundancy of the key K over the $H \times W$ image $I_\circ$ is denoted by ε and is computed as,

$$\varepsilon = \frac{\|K\|}{\|I\|} = \frac{\|K\|}{3HW}, \tag{3.8.198}$$

FPCACEncrypt

```
function [R,Kc,t1,Rs,Rmax,Rmin,Esc,Vsc]=FPCACEncrypt(I,C,m)
%function FPCAC encrypt creates the protected version of I.
%Usage: [R,Kc,t1,Rs,Rmax,Rmin,Esc,Vsc]=FPCACEncrypt(I,C,m);
%Inputs:  I:Original image.
%         C: Cluster number (default: 3).
%         m: Fuzziness  (>1, default 1.05).
%Output: R:Result.
%        Kc:Maps.
%        t1: Elapsed time.
%        Rs:Source indices.
%        Rmax,Rmin:Linear mapping parameters.
%        Esc,Vsc:Expectation vectors and PCA matrices.
%Revision 1.0, Arash Abadpour Spring 2005 (April 16, 2005)
```

Figure 3.73: Implementation of the *FPCACEncrypt* function.

where $\|X\|$ denotes the volume of X in bytes and K contains $[\vec{\rho}_i, V_i]$s and J.

One problem that arises here is that the $\tilde{I}$ computed in (3.8.197) may contain values out of the range of $[0, 255]$. As the result of encryption is saved in one of the color image formats (1 byte per pixel), these truncations produce artifacts in the results of decryption. Thus, we propose to apply a linear mapping on $\tilde{I}$ and to send the parameters of the mapping, which are a bias and a scale, to the decryption stage. Function *BPCAEncrypt* performs the proposed encryption method (see Figure 3.73).

The decryption process is performed in an straightly inverse fashion. Note that, theoretically, the process is completely reversible. We show (by examples) that except for numerical errors, no information is lost during a sequence of encryption–decryption

FPCACDecrypt

```
function [L,t2]=FPCACDecrypt(R,Kc,Rs,Rmax,Rmin,Esc,Vsc)
%function FPCAC decrypt creates the protected version of I.
%Usage: [L,t2]=FPCACDecrypt(R,Kc,Rs,Rmax,Rmin,Esc,Vsc);
%Inputs: R:Input image.
%       Kc:Maps.
%       Rs:Source indices.
%       Rmax,Rmin:Linear mapping parameters.
%       Esc,Vsc:Expectation vectors and PCA matrices.
%Output:  L:Result.
%       t2: Elapsed time.
%Revision 1.0 (c) Arash Abadpour Spring 2005 (April 16, 2005)
```

Figure 3.74: Implementation of the *FPCACDecrypt* function.

processes. Function *BPCADecrypt* performs the proposed encryption method (see Figure 3.74).

Now, lets go back to the reference color descriptors ($[\vec{\rho}_i, U_i]$ for $i = 1, \cdots, C$). In fact, as the proposed visual encryption method is making fake edges, it is very important to alter the color appearance of the image deliberately. We propose to use a bank of homogenous swatches for this purpose (see Figure 3.75). Assume that the bank contains k swatches with the color descriptors of $[\vec{\rho}_i, U_i]$ for $i = 1, \cdots, k$. The correspondence table in Figure 3.72 holds an index to this bank for each cluster. The elements of this table are produced by a proposed matching process which compares two color descriptors. The comparison is performed using a weighted Euclidean distance incorporating the expectation vectors and the polarized version of the PCA

Figure 3.75: Homogenous swatch bank.

matrices discussed in Section 3.1.4.

Doing as such, the distance between a cluster of the image (with expectation vector of $\vec{\eta}$ and FPCA matrix of V) and a homogenous swatch in the bank (with expectation vector of $\vec{\rho}$ and PCA matrix of U) is defined as,

$$\delta^2 = \frac{1}{4\pi^2}\Big[(\theta_V - \theta_U)^2 + (\phi_V - \phi_U)^2 + (\psi_V - \psi_U)^2\Big] + \frac{1}{255^2}\big\|\vec{\eta} - \vec{\eta}\big\|^2. \quad (3.8.199)$$

The $1/4\pi^2$ and $1/255^2$ coefficients are added to map both angles and color values to

the range $[0, 1]$. So, for each cluster, a proper homogenous swatch is selected and its descriptor is used to encrypt the image and is also sent to the decryption process to decrypt the true color vectors.

3.8.2 Block–Based Approach

Assume that the $H \times W$ color image I is given. I is first split into non–overlapping rectangular areas of size $h \times w$, where h and w are given by the user. Here, we put some restrictions for the choice of h and w. The two common compression techniques of JPEG and JPEG2000 work on 8×8 and 32×32 blocks of the given images, respectively. Thus, we urge w and h to be selected in the way that $m \big| w, h$, where $x|y$ means x divides y and m is the block–size of the utilized compression method (apparently, $m = 8$ for JPEG and $m = 32$ for JPEG2000). We will return to the reason for this restriction. As the images in **CIP** dataset are 512×512 images, the user is asked for an integer N in the range of $[0, 9 - \log_2 m]$. Then h and w are selected as $H2^{-N}$ and $W2^{-N}$, respectively. In this, way the image is split into 2^{2N} rectangular regions.

Let's return to the problem in hand. Call the expectation vector of the block I_{ij} as $\vec{\eta}_{ij}$. We propose to use three values of a uniform quantized random variable r as θ_{ij}, ϕ_{ij}, and ψ_{ij} (see Section 3.1.8). These values are produced using the value of A, which is given by the user in the range $[0, 2\pi]$ and the value of $q \in [0, 8]$. We will discuss these parameters and their rule in the results. Now, assume that $V_{ij} \sim (\theta_{ij}, \phi_{ij}, \psi_{ij})$

is computed as discussed in Section 3.1.4. For each color vector $\vec{c}$ in I_{ij}, we compute the new color vector $\vec{d}$ as,

$$\vec{d} = V_{ij}(\vec{c} - \vec{\eta}_{ij}) + \vec{\eta}_{ij}. \tag{3.8.200}$$

This process is performed for all blocks and all pixels within them. The result is the image I_e, the image equivalent to I in its content, but containing deliberately embedded fake edges and color distortions. the parameters for this process are N (number of blocks), A (amplitude), and q (quantization levels). The result of the process is the encrypted image I_e, plus three $2^N \times 2^N$ arrays of q–bit values for storing θ_{ij}, ϕ_{ij}, and ψ_{ij}. As I_e may contain values out of the $[0, 255]$ range, which are cut when I_e is to be saved in a standard format like JPEG, we perform a linear scaling on it. The parameters of the scaling (a bias and a scale) are also saved. Functions *bpcaencrypt* and *bpcacode* encrypt a given image and produce the corresponding ASCII code (see Figure 3.76 and 3.77).

To decrypt an image it is enough to revert the scaling, and then to use the values of θ_{ij}, ϕ_{ij}, and ψ_{ij} to reproduce V_{ij}. Then, using,

$$\vec{c} = V_{ij}^T(\vec{d} - \vec{\eta}_{ij}) + \vec{\eta}_{ij}, \tag{3.8.201}$$

the original image is reconstructed. Note that the expectation vector of the ij–th block in I and I_e are identical. Hence, η_{ij} should not be sent to the decrypter. Also note that, unless for numerical errors and truncations, nothing is lost in this process. We call the image produced by the decrypter as I_d. Also, the intervals elapsed on

BPCAEncrypt

```
function [J,a1,a2,a3,bias,scale,t,sz,comp]=BPCAEncrypt(I,N,NA,Qlevel)
%function block PCA encrypt performs what the name implies.
%Usage:[J,a1,a2,a3,bias,scale,t,sz,comp]=BPCAEncrypt(I,N,NA,Qlevel);
%Inputs: I: Original Image.
%        N: Number of blocks in a row/column.
%        NA: Noise Amplitude.
%        Qlevel: Quantization levels.
%Outputs: J:Result.
%         a1,a2,a3:Angles.
%         bias, scale: Normalizing factors.
%         t:Elapsed time.
%         sz:Key size.
%         comp:Number of possible keys.
%Revision 2.0 Arash Abadpour, Spring 2005 (April 4, 2005)
```

Figure 3.76: Implementation of the *BPCAEncrypt* function.

BPCACode

```
function Y=BPCACode(a1,a2,a3,bias,scale,Np,NAp,NQ)
%function block PCA decrypt performs what the name implies.
%Usage:Y=BPCACode(a1,a2,a3,bias,scale,Np,NAp,NQ)
%Inputs: a1,a2,a3:Angles.
%        bias, scale: Normalizing factors.
%        Np: 2∧Np is the Block Count in a row/column.
%        NAp: 2∧-(NAp) is the noise amplitude.
%        NQ: 2∧NQ is the quantization levels of random codes.
%Outputs: Y:Result.
%Revision 1.0 Arash Abadpour, Spring 2005 (March 31, 2005)
```

Figure 3.77: Implementation of the *BPCACode* function.

BPCADecrypt

```
function [J,t]=BPCADecrypt(I,a1,a2,a3,bias,scale,N,NA,Qlevel)
%function block PCA decrypt performs what the name implies.
%Usage:[J,t]=BPCADecrypt(I,a1,a2,a3,bias,scale,N,NA,Qlevel);
%Inputs: I: Encrypted image.
%        a1,a2,a3:Angles.
%        bias, scale: Normalizing factors.
%        N: Number of blocks in a row/column.
%        NA: Noise Amplitude.
%        Qlevel: Quantization levels.
%Outputs: J:Result.
%        t:Elapsed time.
%Revision 2.0 Arash Abadpour, Spring 2005 (April 4, 2005)
```

Figure 3.78: Implementation of the *BPCADecrypt* function.

performing encryption and decryption are called t_e and t_d, respectively. Figure 3.79 shows a flowchart of the proposed method. Function *bpcadecrypt* shown in Figure 3.78 performs decryption.

When an image I is encrypted into the image I_e, it should be exposed to the consumers. Hence, a standard image format should convey the contents of I_e. As the dynamic range of I_e is changed to $[0, 255]$, saving I_e as a BMP file only results in truncations, which is an acceptable distortion (gives a PSNR of $54dB$). The situation is more crucial when I_e is going to be saved is a more professional format like JPEG or JPEG2000. The proposed method well tolerates this situation. In fact I_e contains consistent blocks of size $h \times w$, and the inconsistencies only occur at the edges of these

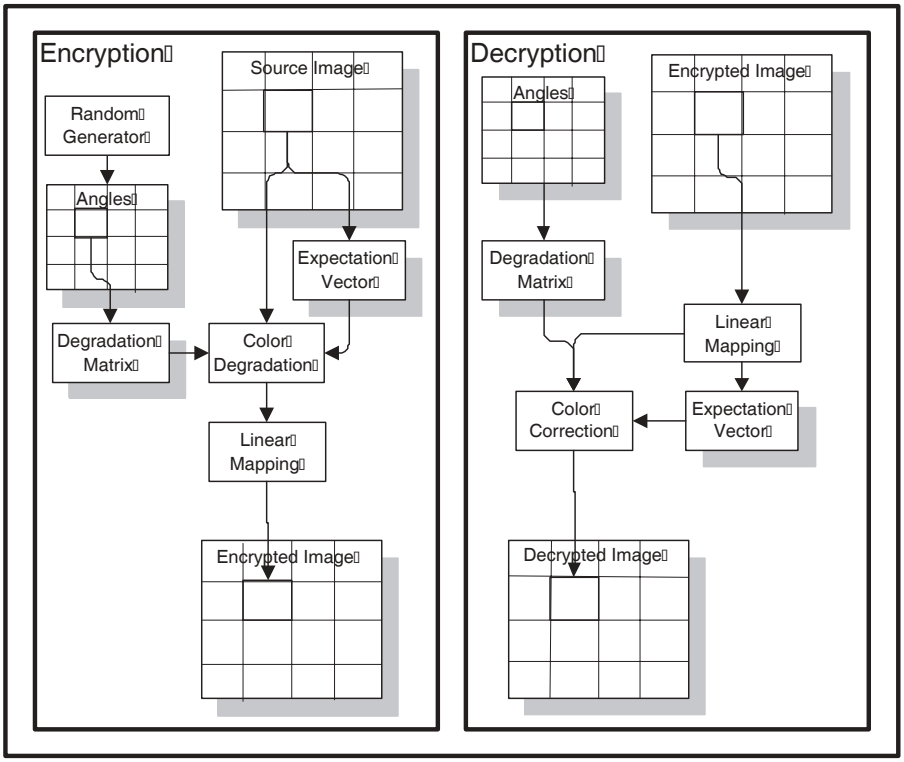


Figure 3.79: Flowchart of the proposed copy protection method.

blocks. In accordance, JPEG and JPEG2000 divide an image into non–overlapping blocks and perform corresponding transform on these blocks, separately. Hence, any non–stationarity caused within a block will result in a low–compression rate or low–quality event in the result of compression. By retaining the $m|h, w$ condition, the process makes sure that no non–stationarity occurs inside a block. Hence, I_e may be fed to a JPEG–based compression module with no aftermath on the quality of I_d, except for the losses produced by the compression itself. We emphasize that neglecting the $m|w, h$ condition results in sever losses. One important point here is that, in the JPEG standard there is the possibility to down–sample the chrominance planes by half or one–fourth. Using this option will result in a fake grid in I_d and should be prohibited.

3.9 Color Transfer

Although, color transfer between images is a new field, it has gathered attention of researchers. In this section, first a PCA–based colorizing method is discussed in Section 3.9.1. Then, two different approaches of supervised and unsupervised recoloring are discussed in Sections 3.9.2 and 3.9.3, respectively.

3.9.1 Colorizing

The proposed method for grayscale image colorization does not depend on any particular color space and works using a linear definition for the achromatic norm defined

below. In Section 3.9.1 some considerations for a suitable norm function are discussed. The proposed method is presented in Section 3.9.1.

Color to Grayscale Transformation

Many types of illumination measures are discussed in the literature (also called brightness and intensity), with different meanings and aims (*e.g.*, see [273, 350, 138, 328, 58, 5]). To be general, we call such a measure the achromatic norm of a color vector or its norm. To use the linear algebra methods we limit the work to linear norm functions generally shown as:

$$\langle \vec{v} \rangle = \vec{N}^T \vec{v}, \tag{3.9.202}$$

where the vector $\vec{N}$ is selected in the way that it satisfies the condition $\langle \vec{1} \rangle = 1$. The above–defined norm function satisfies two simple but important conditions of:

$$\langle \alpha \vec{v}_1 + \vec{v}_2 \rangle = \alpha \langle \vec{v}_1 \rangle + \langle \vec{v}_2 \rangle, \tag{3.9.203}$$

$$E\{\langle \vec{x} \rangle\} = \langle E\{\vec{x}\} \rangle. \tag{3.9.204}$$

The proofs may be obtained by the mentioned definitions and a little linear algebra.

Grayscale Image Colorizing

As the transformation from a 3–D color vector to a single intensity number is irreversible, there can be many (extremely) different colors with the same appearance in the grayscale image. One method to distinguish such spurious points is to use some

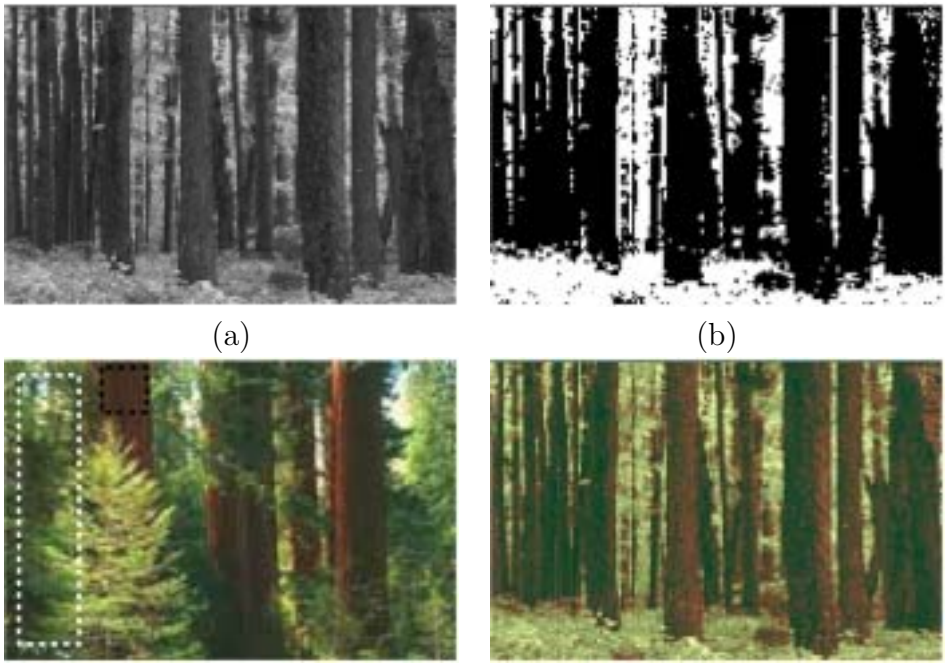
neighborhood information, as proposed in *Welsh et al.*'s method [375]. Their method is based on an unprovable assumption that the color regions of the same object lie in the same range of intensity. Having in mind the desperate performance that they have reported, along with the wide range of images in which the method fails (containing the major category of face images), the need for an efficient colorizing method arises. Furthermore, they combine the two segmentation stages for the reference and the source images, while these are in entirely different domains (one is a color image whilst the other is essentially a grayscale one). To overcome the shortcomings of *Welsh et al.*'s work [375], here we separate the color extraction and segmentation processes.

As discussed in Section 2.2.14, a color transfer method includes both a segmentation phase (the classifier) and a color generating process. While most researchers have focused on the segmentation task and have left the suitability of color generating tasks with less attention, we emphasize on the color generation stage. As discussed in Section 2.2.14, the color generation process in all available colorizing works includes transferring (sometimes an interpolated) set of color components computed in one of the color spaces. Due to the inefficiency of the color spaces in decorrelating the color components (see Section 2.1), we propose an entirely different approach.

Here we assume that regarding the problem in hand, the source image is separated into a few segments describing the content of the points in the physical phenomenon. Many examples of such segmentation processes are available in the literature (for comprehensive surveys see [144, 167, 276]). We emphasize that the efficiency of the segmentation stage massively influences the overall performance of the color transfer method, but here we have only focused on developing proper color extraction/generation algorithms (many of the available color transfer works can be easily adopted to work with the color processing methods proposed in this thesis).

Hence, we propose a colorizing approach which asks the user for an indexed map which for each pixel gives the corresponding class to which it belongs. Then for each class, the user gives a homogenous color swatch as the reference color from which the color information is extracted and added to the respective segments of the source image. The process runs on all swatches to produce the destination image. Figure 3.80 shows a sample set of inputs to the proposed colorizing method along with the result. Figure 3.80–a shows the source image categorized into into classes of eaves and bark as shown in Figure 3.80–b. Figure 3.80–c illustrates the reference image and the two representative swatches and Figure 3.80–d shows the result of the proposed colorizing method.

As discussed in Section 3.3 the expectation vector and the direction of the first principal component of a given cluster are proper descriptors for modelling its color



(a) (b)

(c) (d)

Figure 3.80: Typical input and output of the proposed colorizing method. (a) Source image. (b) Index map. (c) Reference image and the representative swatches. (d) Result.

content. Having a grayscale pixel and the above color descriptors we propose to generate a proper color vector using the color descriptors of the reference swatch. As there exists no objective criterion for comparing different colorizing algorithms in the literature, here we propose three conditions to be met by a perfect colorizing process. Firstly, it is reasonable to expect a colorizing process to not to alter the original grayscale information of the image. Thus, For the source image I_1, the reference image I_2, and the destination image I_1', we expect the grayscale version of I_1' to be identical to I_1. This ensures that converting the destination image back to the grayscale representation reverts it to the source image. As the second and the third conditions we force the colorizing method to embed the same color information extracted from the reference image into the destination image. Thus, the transfer scheme should be designed in the way that using the destination as the reference image for another colorizing task results the same as using the original reference image. We formulate the second and the third conditions as $\vec{\eta}_1' \parallel \vec{\eta}_2$ and $\vec{v}_1' = \vec{v}_2$, where $\vec{x} \parallel \vec{y}$ means $\vec{x}$ is parallel to $\vec{y}$. The first condition forces the expectation vector of the colorized image to be parallel to the expectation vector of the reference image and the third condition limits the direction of its first principal component to be identical with the references one. Note that by expectation vector we mean the vector corresponding to a single segment or a single swatch. Note that nothing is said about segments in a color image containing a set of non–homogenous color vectors.

We will come back to these conditions and discuss them more later.

In this work, we propose a new category of colorizing methods, that generate the color vector corresponding to the grayscale c_1 as a function of η_1, $\vec{\eta}_2$, and $\vec{v}_2$. These notions correspond to the expectation of the grayscale values in the source image and the expectation vector and the direction of the first principal component of the associated color cluster in the reference image. Note that, using the segmentation information given by the user, all variables relate to the same cluster to which the pixel with the grayscale value of c_1 in the source image is related. The color vector relating to c_1 is generated as,

$$\vec{c}_1' = \Phi_{\eta_1,\vec{\eta}_2,\vec{v}_2}(c_1), \tag{3.9.205}$$

where, Φ should satisfy the above three constraints. Many different formulations are designed and implemented by the author. The two most appropriates are,

$$\vec{c}_1' = \vec{\eta}_2 + \big(c_1 - \langle\vec{\eta}_2\rangle\big)\frac{\vec{v}_2}{\langle\vec{v}_2\rangle}, \tag{3.9.206}$$

$$\vec{c}_1' = \frac{\eta_1}{\langle\vec{\eta}_2\rangle}\vec{\eta}_2 + (c_1 - \eta_1)\frac{\vec{v}_2}{\langle\vec{v}_2\rangle}. \tag{3.9.207}$$

Lets give some clues about the origin of (3.9.206). Assume that the following equation is meant to give a proper colorizing,

$$\vec{c}_1' = f(c_1)\vec{\eta}_2 + g(c_1)\vec{v}_2. \tag{3.9.208}$$

Here, $\vec{c}_1'$ is the colorized version of c_1 according to the color information in the pair $\vec{\eta}_2$ and $\vec{v}_2$. Here, we will discuss the choices for $f : R \to R$ and $g : R \to R$ given the sets of constraints.

The set of colorized vectors should satisfy:

$$\langle \vec{c}_1' \rangle = c_1, \tag{3.9.209}$$

$$\vec{v}_1' = \vec{v}_2, \tag{3.9.210}$$

$$\vec{\eta}_1' - \vec{\eta}_2 \| \vec{v}_2. \tag{3.9.211}$$

Note that here we may assume that $\vec{\eta}_2 \perp \vec{v}_2$. In the contrary case, one may easily replace $\vec{\eta}_2$ with $\vec{\eta}_2 - \vec{v}_2^T \vec{\eta}_2 \vec{v}_2$. From (3.9.209) we have,

$$\langle \vec{c}_1' \rangle = f(c_1)\langle \vec{\eta}_2 \rangle + g(c_1)\langle \vec{v}_2 \rangle = c_1 \to g(c_1) = \frac{1}{\langle \vec{v}_2 \rangle}\Big[c_1 - \langle \vec{\eta}_2 \rangle f(c_1)\Big], \tag{3.9.212}$$

converting the general formulation in (3.9.208) into,

$$\vec{c}_1' = f(c_1)\vec{\eta}_2 + \frac{1}{\langle \vec{v}_2 \rangle}\Big[c_1 - \langle \vec{\eta}_2 \rangle f(c_1)\Big]\vec{v}_2. \tag{3.9.213}$$

Note that,

$$\vec{\eta}_1' = E\{\vec{c}_1'\} = E\{f(c_1)\}\vec{\eta}_2 + \frac{1}{\langle \vec{v}_2 \rangle}\Big[E\{c_1\} - \langle \vec{\eta}_2 \rangle f(c_1)\}\Big]\vec{v}_2. \tag{3.9.214}$$

Comparing (3.9.211) with (3.9.214) yields,

$$E\{f(c_1)\} = 1 \to f(c_1) = 1 + h(c_1), E\{h(c_1)\} = 0. \tag{3.9.215}$$

Lets work on satisfying the condition in (3.9.210). First note that,

$$\vec{c}_1' - \vec{\eta}_1' = h(c_1)\vec{\eta}_2 + \frac{1}{\langle \vec{v}_2 \rangle}\Big[(c_1 - E\{c_1\}) - \langle \vec{\eta}_2 \rangle h(c_1)\Big]\vec{v}_2. \tag{3.9.216}$$

First we prove a theorem. Assume that $2n$ real values of $\alpha_1, \cdots, \alpha_n$ and $\beta_1, \cdots, \beta_n$ are given. Also assume that non–zero vectors of $\vec{v}_1 \perp \vec{v}_2$ are given. We claim that $\vec{v}_1$ is the principal direction of $\vec{x}_i = \alpha_i \vec{v}_1 + \beta_i \vec{v}_2, i = 1, \cdots, n$ if and only if $\sigma_{\alpha\beta}$ is zero and that $\sigma_\alpha^2 > \sigma_\beta^2$. To prove this theorem first note that,

$$C = \frac{1}{n}\sum_{i=1}^{n}\Big[(\alpha_i - \bar{\alpha})\vec{v}_1 + (\beta_i - \bar{\beta})\vec{v}_2\Big]\Big[(\alpha_i - \bar{\alpha})\vec{v}_1 + (\beta_i - \bar{\beta})\vec{v}_2\Big]^T, \tag{3.9.217}$$

where $\bar{\alpha}$ and $\bar{\beta}$ are the expectation of α_i andβ_i, respectively. Deriving (3.9.218) we have,

$$C = \sigma_\alpha^2 \vec{v}_1 \vec{v}_1^T + \sigma_\beta^2 \vec{v}_2 \vec{v}_2^T + \sigma_{\alpha\beta}\Big[\vec{v}_1 \vec{v}_2^T + \vec{v}_2 \vec{v}_1^T\Big]. \tag{3.9.218}$$

if $\vec{v}_1$ is an eigenvector of C then we should have,

$$C\vec{v}_1 = \sigma_\alpha^2 \vec{v}_1 + \sigma_{\alpha\beta}\vec{v}_2 = \lambda_1 \vec{v}_1 \rightarrow \sigma_{\alpha\beta} = 0. \tag{3.9.219}$$

Note that in this situation $\vec{v}_1$ and $\vec{v}_2$ are the eigenvectors of C according to the eigenvalues of σ_α^2 and σ_β^2. Now (3.9.218) turns into,

$$C = \sigma_\alpha^2 \vec{v}_1 \vec{v}_1^T + \sigma_\beta^2 \vec{v}_2 \vec{v}_2^T = \begin{pmatrix} v_{11} & v_{21} & 0 \\ v_{12} & v_{22} & 0 \\ v_{13} & v_{23} & 0 \end{pmatrix} \begin{pmatrix} \sigma_\alpha^2 & 0 & 0 \\ 0 & \sigma_\beta^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & c_{13} \\ v_{21} & v_{22} & c_{23} \\ 0 & 0 & 0 \end{pmatrix}, \tag{3.9.220}$$

which is the SVD transform of C, proving that $Rank(C) = 2$. Hence, the only eigenvectors of C are $\vec{v}_1$ and$\vec{v}_2$. Accordingly if $\sigma_\alpha^2 > \sigma_\beta^2$ then $\vec{v}_1$ would be the eigenvector corresponding to the largest eigenvalue (the principal direction).

Lets return to the original problem. The equation in (3.9.216) is a special case of the theorem discussed at the above. Hence, for $\vec{v}_2$ becoming the principal direction of $\vec{c}_1^{\,\prime}$ we should have,

$$E\Big\{h(c_1)\Big[(c_1 - E\{c_1\}) - \langle\vec{\eta}_2\rangle h(c_1)\Big]\Big\} = 0, \tag{3.9.221}$$

$$\frac{1}{\langle\vec{v}_2\rangle^2}E\Big\{\Big[(c_1 - E\{c_1\}) - \langle\vec{\eta}_2\rangle h(c_1)\Big]^2\Big\} \gg E\Big\{h(c_1)^2\Big\}. \tag{3.9.222}$$

Note that (3.9.221) turns into,

$$\langle\vec{\eta}_2\rangle E\Big\{h(c_1)^2\Big\} = E\Big\{c_1 h(c_1)\Big\}, \tag{3.9.223}$$

and (3.9.222) turns into,

$$\sigma_{c_1}^2 \gg \Big[\langle\vec{\eta}_2\rangle^2 + \langle\vec{v}_2\rangle^2\Big]E\Big\{h(c_1)^2\Big\}. \tag{3.9.224}$$

As such, for,

$$\vec{c}_1^{\,\prime} = \Big[1 + h(c_1)\Big]\vec{\eta}_2 + \frac{1}{\langle\vec{v}_2\rangle}\Big[c_1 - \langle\vec{\eta}_2\rangle[1 + h(c_1)]\Big]\vec{v}_2. \tag{3.9.225}$$

being a colorizing transform complying with (3.9.209), (3.9.210), and (3.9.211), $h : R \to R$ should satisfy,

$$\begin{cases} E\{h(c_1)\} = 0 \\ \langle\vec{\eta}_2\rangle E\Big\{h(c_1)^2\Big\} = E\Big\{c_1 h(c_1)\Big\} \\ \sigma_{c_1}^2 \gg \Big[\langle\vec{\eta}_2\rangle^2 + \langle\vec{v}_2\rangle^2\Big]E\Big\{h(c_1)^2\Big\} \end{cases}. \tag{3.9.226}$$

For example, lets investigate the case of $h(c_1) = \alpha c_1 + \beta$. Checking the three conditions stated in (3.9.226) proves that the only choice is $h(c_1) \equiv 0$. This will lead

to the colorizing process described as,

$$\vec{c}_1' = \vec{\eta}_2 + \Big[c_1 - \langle\vec{\eta}_2\rangle\Big]\frac{\vec{v}_2}{\langle\vec{v}_2\rangle}, \tag{3.9.227}$$

which is the same as (3.9.206).

Investigating the fulfillment of the first condition in (3.9.206) we have,

$$\langle\vec{c}_1'\rangle = \langle\vec{\eta}_2\rangle + \big(c_1 - \langle\vec{\eta}_2\rangle\big)\frac{\langle\vec{v}_2\rangle}{\langle\vec{v}_2\rangle} = c_1, \tag{3.9.228}$$

which means turning the destination vector back to the grayscale domain reverts it to the same grayscale value as it is in the source image. Also, we have,

$$\vec{\eta}_1' = E\{\vec{c}_1'\} = \vec{\eta}_2 + \big(\eta_1 - \langle\vec{\eta}_2\rangle\big)\frac{\vec{v}_2}{\langle\vec{v}_2\rangle}, \tag{3.9.229}$$

which is not parallel to $\vec{\eta}_2$, contradicting the second condition. To investigate the third condition we compute the covariance matrix of $\vec{c}_1'$ as,

$$C_1' = E\{(\vec{c}_1' - \vec{\eta}_1')(\vec{c}_1' - \vec{\eta}_1')^T\}. \tag{3.9.230}$$

Note that,

$$\vec{c}_1' - \vec{\eta}_1' = \vec{\eta}_2 + \big(c_1 - \langle\vec{\eta}_2\rangle\big)\frac{\vec{v}_2}{\langle\vec{v}_2\rangle} - \vec{\eta}_2 - \big(E\{c_1\} - \langle\vec{\eta}_2\rangle\big)\frac{\vec{v}_2}{\langle\vec{v}_2\rangle} = \big(c_1 - \eta_1\big)\frac{\vec{v}_2}{\langle\vec{v}_2\rangle} \tag{3.9.231}$$

Substituting (3.9.231) in (3.9.230) gives,

$$C_1' = \frac{E\{(c_1 - \eta_1)^2\}}{\langle\vec{v}_2\rangle^2}\vec{v}_2\vec{v}_2^T = \frac{\sigma_{c_1}^2}{\langle\vec{v}_2\rangle^2}\vec{v}_2\vec{v}_2^T. \tag{3.9.232}$$

Also, we have,

$$\frac{\sigma_{c_1}^2}{\langle\vec{v}_2\rangle^2}(\vec{v}_2\vec{v}_2^T)\vec{v}_2 = \frac{\sigma_{c_1}^2\|\vec{v}_2\|^2}{\langle\vec{v}_2\rangle^2}\vec{v}_2 = \frac{\sigma_{c_1}^2}{\langle\vec{v}_2\rangle^2}\vec{v}_2, \tag{3.9.233}$$

which means that the vector $\vec{v}_2$ is an eigenvector of the matrix C_1'. As the rank of the 3×3 matrix C_1' is unity, $\vec{v}_2$ is its only eigenvector. Thus, the direction of the first principal component of $\vec{c}_1'$ is the same as the ones relating to the reference image (fulfilling the third condition). Hence, (3.9.206) satisfies the first and the third criteria while failing the second.

Returning to the solution proposed in (3.9.207) we have,

$$\langle \vec{c}_1' \rangle = \frac{\eta_1}{\langle \vec{\eta}_2 \rangle} \langle \vec{\eta}_2 \rangle + (c_1 - \eta_1) \frac{\langle \vec{v}_2 \rangle}{\langle \vec{v}_2 \rangle} = c_1, \tag{3.9.234}$$

meaning that turning back the destination image to the grayscale domain reverts it back to the source image. The second criterion uses $E\{\vec{c}_1'\}$ where we have,

$$E\{\vec{c}_1'\} = \frac{\eta_1}{\langle \vec{\eta}_2 \rangle} \vec{\eta}_2 + (\eta_1 - \eta_1) \frac{\vec{v}_2}{\langle \vec{v}_2 \rangle} = \frac{\eta_1}{\langle \vec{\eta}_2 \rangle} \vec{\eta}_2 \Big\| \vec{\eta}_2, \tag{3.9.235}$$

complying with the second criterion. For the third criterion we compute the covariance matrix of $\vec{c}_1'$ as,

$$C_1' = E\{(\vec{c}_1' - \vec{\eta}_1')(\vec{c}_1' - \vec{\eta}_1')^T\}. \tag{3.9.236}$$

Also, we have,

$$\vec{c}_1' - \vec{\eta}_1' = \frac{\eta_1}{\langle \vec{\eta}_2 \rangle} \vec{\eta}_2 + (c_1 - \eta_1) \frac{\vec{v}_2}{\langle \vec{v}_2 \rangle} - \frac{\eta_1}{\langle \vec{\eta}_2 \rangle} \vec{\eta}_2 = (c_1 - \eta_1) \frac{\vec{v}_2}{\langle \vec{v}_2 \rangle}, \tag{3.9.237}$$

which results like (3.9.232) proving that the third criterion is also fulfilled. Thus, the formula in (3.9.207) complies with all three criteria.

Considering the computation costs, both (3.9.206) and (3.9.207) cost three subtractions, three additions and three multiplications for each pixel in the colorization stage.

As the method by *Welsh et al.* [375] is adopted in many other works, a subjective test is performed to compare the performance of that approach with the proposed method. To do so, 26 persons at the ages of 17 to 27 were asked to reorder the three images of *Welsh*'s [375] synthesized images and the results of the methods proposed in (3.9.206) and (3.9.207), in terms of goodness [189] (Table 3.3). The subjects were unaware of the algorithms but were literate people. Neither the source image nor the reference images were exposed to the subjects and they were able to scroll the monitor in order to watch any of the three images at any time they intended to see, but not zooming through them. The images were shown in a randomly ordered fashion and no timing threshold is set. The answers were given as strings containing "1","2", and "3" along with "," and "/". For example, "1/2,3" denotes that the first image is the best, and the second and the third are the same in the next place. Subjects were not allowed to contribute the results. The orders were changed to scores in the range of $[0 \dots 3]$ and subsequently an statistical analysis is preformed on them. Also, human perception is measured by means of impairment (Table 3.2). The results are discussed in Section 4.10.1.

Table 3.2: Impairment scales [189].

Scale	**Meaning**
1	Not noticeable.
2	Just noticeable.
3	Definitely noticeable but only slight impairments.
4	Impairment not objectionable.
5	Somewhat objectionable.
6	Definitely objectionable.
7	Extremely objectionable.

Table 3.3: Goodness scales [189].

Scale	**Meaning**
5	Excellent
4	Good
3	Fair
2	Poor
1	Unsatisfactory

3.9.2 Region–Based Recoloring

User contribution in the proposed color image recoloring method is limited to selecting a few corresponding swatches in the reference and the source images, along with tuning a one–parameter membership function. In Section 3.9.2, we first propose a framework to transfer the color content of a homogenous swatch to another homogenous swatch and discuss the mathematical perspective. We call this intermediate stage single–swatch recoloring. Then, using the fuzzification method (discussed in section 3.4.1) we generalize the method to recolor the entire image using a set of corresponding homogenous swatches (notations in this Section are the same as in Section 3.9.1).

Figure 3.81 shows a sample result of the proposed recoloring method. Figure 3.81–a and Figure 3.81–b show the source and the reference images with the overlayed swatches. Figure 3.81–c and Figure 3.81–d illustrate the fuzzy membership of the pixels of the source images to the first and the second swatches of the source image, respectively. Figure 3.81–e and Figure 3.81–f illustrate the results of single swatch recoloring. Note that in Figure 3.81–e the flower color has changed to the desired one, while the leaves are recolroized perfectly in Figure 3.81–f. Finally, Figure 3.81–g shows the final result of the proposed method, incorporating both the single–swatch recoloring results and the fuzzy membership values.

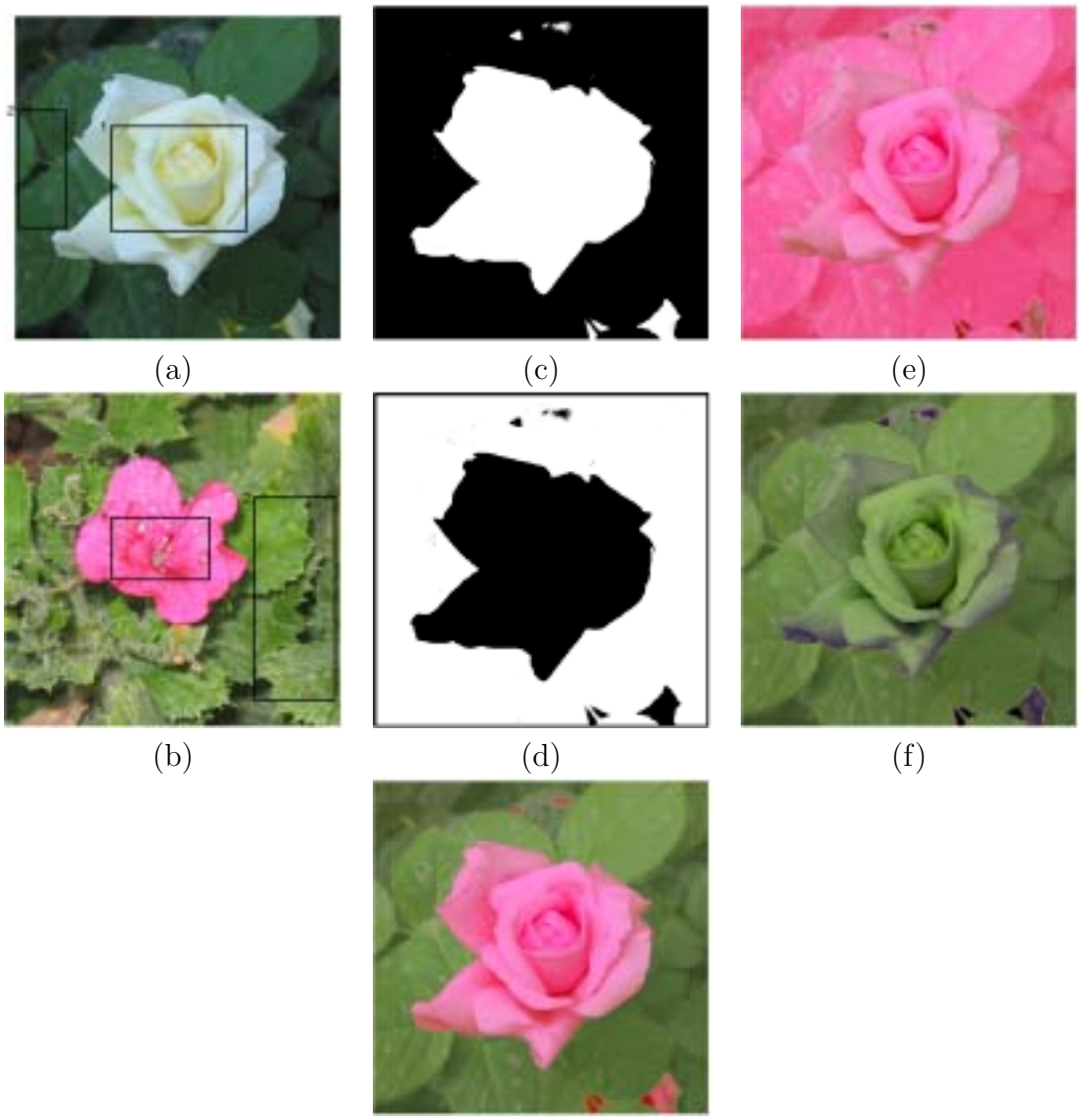


Figure 3.81: Sample result of the proposed recoloring method. (a) Source Image and the overlayed swatches. (b) Reference image and the overlayed swatches. (c) Fuzzy membership of source pixels to the flower group. (d) Fuzzy membership of source pixels to the leaf group. (e) Results of the proposed single–swatch recoloring according to the flower swatches. (f) Results of the proposed single–swatch recoloring according to the leaf swatches. (g) Final result.

Single–Swatch Recoloring

Assume that the source image I_1 and the reference image I_2 both contain homogenous vectors. This assumption is the same as *Reinhard et al.*'s first recoloring method [297]. As discussed in Section 3.3 the PCA framework is a reliable tool to model color distribution of homogenous swatches. Thus, here we use a linear method to propose the single–swatch recoloring method.

Assume that the vectors $\vec{\eta}_1$, $\vec{\eta}_2$, V_1, and V_2 denote the expectation vector of color information in I_1 and I_2 and their corresponding PCA matrices, respectively. The PCA matrix is the one containing the eigenvectors of the respective covariance matrix as its columns sorted by the eigenvalues in a descending fashion. Note that both V_1 and V_2 are orthonormal matrices. Here, we propose to compute the color vector $\vec{c}_1'$ as the result of recoloring the color vector $\vec{c}_1$ as,

$$\vec{c}_1' = V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1) + \vec{\eta}_2. \tag{3.9.238}$$

Now, lets investigate the properties of $\vec{c}_1'$ in more details. Computing the expectation vector of I_1' (the destination image) we have,

$$\vec{\eta}_1' = E\{\vec{c}_1'\} = V_2 V_1^{-1} E\{\vec{c}_1 - \vec{\eta}_1\} + \vec{\eta}_2 = \vec{\eta}_2. \tag{3.9.239}$$

Note that V_1 and V_2 are constant matrices. Thus, the expectation vector of the destination image is identical to the expectation vector of the reference image. For

C_1' (the covariance matrix of I_1') we have,

$$C_1' = E\{(\vec{c}_1' - \vec{\eta}_1')(\vec{c}_1' - \vec{\eta}_1')^T\}. \tag{3.9.240}$$

Also, we have,

$$\vec{c}_1' - \vec{\eta}_1' = V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1) + \vec{\eta}_2 - \vec{\eta}_2 = V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1). \tag{3.9.241}$$

Substituting (3.9.241) in (3.9.240) results in,

$$C_1' = E\left\{ V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)\Big(V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)\Big)^T \right\} = \tag{3.9.242}$$
$$E\Big\{ V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)(\vec{c}_1 - \vec{\eta}_1)^T V_1 V_2^{-1} \Big\} = V_2 V_1^{-1} C_1 V_1 V_2^{-1},$$

where, C_1 is the covariance matrix of I_1. Assume that $\vec{v}$ is the eigenvector of C_1 corresponding to the eigenvalue of λ. Thus, we have $C_1\vec{v} = \lambda\vec{v}$. Consider the equality of,

$$C_1' V_2 V_1^{-1}\vec{v} = V_2 V_1^{-1} C_1 V_1 V_2^{-1} V_2 V_1^{-1}\vec{v} = V_2 V_1^{-1} C_1 \vec{v} = \lambda V_2 V_1^{-1}\vec{v}. \tag{3.9.243}$$

Substituting,

$$\vec{u} = V_2 V_1^{-1}\vec{v}, \tag{3.9.244}$$

in (3.9.243) we have $C_1'\vec{u} = \lambda\vec{u}$. Thus, $\vec{u}$ is the eigenvector of C_1' with the same eigenvalue of λ. Hence, the eigenvectors of C_1' are computed from those of C_1 using (3.9.244). Note that their order remains identical. This proves that V_1' (the PCA matrix of I_1') equals,

$$V_1' = V_2 V_1^{-1} V_1 = V_2. \tag{3.9.245}$$

Equations (3.9.245) and (3.9.239) prove that when recoloring a source image with a reference image using the proposed method, the color models get identical. This important result has a few fascinating outcomes. Firstly, assume that after recoloring I_1 with I_2 to have I_1' we are not satisfied with it and we want to revert it. Recoloring I_1' with the color information extracted from the original source image, the new destination image (I_1'') is computed as,

$$\vec{c}_1'' = V_1 V_2^{-1}(\vec{c}_1' - \vec{\eta}_2) + \vec{\eta}_1 = V_1 V_2^{-1}(V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1) + \vec{\eta}_2 - \vec{\eta}_2) + \vec{\eta}_1 = \vec{c}_1. \quad (3.9.246)$$

Thus, I_1'' is identical to the original I_1, meaning that the proposed single–swatch recoloring scheme is completely reversible.

As another property of the proposed recoloring method, consider the three images I_1, I_2, and I_3. Assume that we have performed recoloring process on I_1 using the reference image of I_2 to acquire I_1'. Also, assume that we have used I_1' as a new source image and I_3 as the reference image to compute I_2'. Also, assume eliminating the intermediate stage and recolor I_1 according to I_3 to get I_1''. Thus, we have,

$$\vec{c}_2' = V_3 V_2^{-1}(\vec{c}_1' - \vec{\eta}_2) + \vec{\eta}_3 = V_3 V_2^{-1}(V_2 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1) + \vec{\eta}_2 - \vec{\eta}_2) + \vec{\eta}_3 = V_3 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1) + \vec{\eta}_3, \quad (3.9.247)$$

and,

$$\vec{c}_1'' = V_3 V_1^{-1}(\vec{c}_1 - \vec{\eta}_1) + \vec{\eta}_3. \quad (3.9.248)$$

Comparing (3.9.247) and (3.9.248) proves that I_1'' is identical to I_2'. Hence, the proposed single–swatch recoloring scheme is transitive.

As the final property of the proposed single–swatch recoloring method, consider its effects on the LPRE distance. Having the source image I_1, the reference image I_2, and the destination image I_1', the LPRE distance of c_1 to I_1 is computed as,

$$\tau_{I_1}(\vec{c}_1) = \|\vec{v}_1^T(\vec{c}_1 - \vec{\eta}_1)\vec{v}_1 - (\vec{c}_1 - \vec{\eta}_1)\|. \tag{3.9.249}$$

Note that all images are considered to be homogenous in this stage. Performing the same destination on the destination image we have,

$$\tau_{I_1'}(\vec{c}_1') = \|\vec{v}_1'^T(\vec{c}_1' - \vec{\eta}_1')\vec{v}_1' - (\vec{c}_1' - \vec{\eta}_1')\|. \tag{3.9.250}$$

Using (3.9.238) we have,

$$\vec{c}_1' - \vec{\eta}_2 = V_2V_1^{-1}(\vec{c}_1 - \vec{\eta}_1). \tag{3.9.251}$$

Substituting (3.9.251) in (3.9.250) gives,

$$\tau_{I_1'}(\vec{c}_1') = \|\vec{v}_2^T V_2V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)\vec{v}_2 - V_2V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)\|. \tag{3.9.252}$$

As $\vec{v}_2$ is the first column of V_2, and due to the equality in (3.9.245), we have,

$$\vec{v}_2 = V_2V_1^{-1}\vec{v}_1. \tag{3.9.253}$$

Hence,

$$\tau_{I_1'}(\vec{c}_1') = \|(V_2V_1^{-1}\vec{v}_1)^T V_2V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)(V_2V_1^{-1}\vec{v}_1) - V_2V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)\| = \quad (3.9.254)$$

$$\|\vec{v}_1^T(\vec{c}_1 - \vec{\eta}_1)(V_2V_1^{-1}\vec{v}_1) - V_2V_1^{-1}(\vec{c}_1 - \vec{\eta}_1)\| =$$

$$\left\| V_2V_1^{-1}\left[\vec{v}_1^T(\vec{c}_1 - \vec{\eta}_1)\vec{v}_1 - (\vec{c}_1 - \vec{\eta}_1)\right] \right\|.$$

As both V_1 and V_2 are orthonormal matrices, the $V_2V_1^{-1}$ matrix will be an orthonormal matrix as well. Thus, $V_2V_1^{-1}$ reserves the length. Hence, (3.9.254) results in,

$$\tau_{I_1'}(\vec{c}_1') = \tau_{I_1}(\vec{c}_1). \quad (3.9.255)$$

It directly results that the LPRE homogeneity criterion (3.3.141), the normalized LPRE (3.3.142), and the proposed fuzzification scheme (see Section 3.4.1), are all reserved under the proposed single–swatch recoloring.

As a result of the above mentioned derivations, we proposed a single–swatch recoloring method that,

- Recolors a homogenous swatch according to another homogenous swatch.
- Depends on no parameters.
- After the recoloring process, using the destination image as the reference image in another recoloring task results the same as using the original reference image. It means that the reference and the destination image are identical in color content.

- Is completely reversible.
- Is transitive.
- Reserves LPRE distance, LPRE homogeneity criterion, LPRE normalized distance, and the LPRE–based fuzzification.
- Is fast.

Query Region–Based Recoloring

Consider the source image I_1 and the reference image I_2. Also, assume that we are given the set of homogenous swatches $r_{11}, \cdots, r_{1n}$ in the source image and the set of homogenous swatches $r_{21}, \cdots, r_{2n}$ in the reference image. The problem is to compute the destination image in the way that the transformed version of r_{1i} looks like r_{2i}, for $i = 1, \cdots, n$.

Note that when using r_{1i} and r_{2i} as the source and the reference images to recolor the color vector $\vec{c}_1$, the equation in (3.9.238) gives $\vec{c}'_{1i}$ (see Figure 3.81). In fact, $\vec{c}'_{1i}$ is the recolorized vector given that $\vec{c}_1$ is entirely belonging to r_{1i}. In this stage, we incorporate the fuzzyfication scheme proposed in Section 3.4.1 to blend the results,

$$\vec{c}'_1 = \frac{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)\vec{c}'_{1i}}{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)} = \frac{1}{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)} \sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1) \left[V_{2i} V_{1i}^{-1} (\vec{c}_1 - \vec{\eta}_{1i}) + \vec{\eta}_{2i} \right], \tag{3.9.256}$$

where, V_{1i}, $\vec{\eta}_{1i}$, V_{2i}, and $\vec{\eta}_{2i}$ denote the PCA matrix and the expectation vector of r_{1i}

and r_{2i}, respectively. Deriving (3.9.256) we have,

$$\vec{c}_1' = \frac{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1) V_{2i} V_{1i}^{-1}}{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)} \vec{c}_1 + \frac{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)[\vec{\eta}_{2i} - V_{2i} V_{1i}^{-1} \vec{\eta}_{1i}]}{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)}, \tag{3.9.257}$$

or,

$$\vec{c}_1' = V \vec{c}_1 + \vec{\eta}, \tag{3.9.258}$$

where,

$$V = \frac{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1) V_{2i} V_{1i}^{-1}}{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)}, \tag{3.9.259}$$

$$\vec{\eta} = \frac{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)[\vec{\eta}_{2i} - V_{2i} V_{1i}^{-1} \vec{\eta}_{1i}]}{\sum_{i=1}^{n} h_{r_{1i},\beta}(\vec{c}_1)}. \tag{3.9.260}$$

This new notation shows that the proposed recoloring process a is linear transform. For each pixel the destination vector is computed using the weighted sum of single swatch recoloring matrices and bias vectors. Function *recolor* performs the proposed method (see Figure 3.82).

3.9.3 Clustering–based Recoloring

Assume that according to the given source image I_1 and the given reference image I_2, we have to produce the destination image I_1'. Figure 3.83 shows the flowchart of the proposed recoloring method. Firstly, both images are fed to the FPCAC (see Section 3.5.2). The results of the clustering are the two sets of membership maps $J_{11}, \cdots, J_{1c}$ and $J_{21}, \cdots, J_{2c}$, describing the membership of each pixel in the

recolor

```
function [R,F,t]=recolor(Is,Ir,Rs,Rr,Os,Or,W,a,b);
%This function performs the recoloring. For more information refer to,
%A. Abadpour and S. Kasaei, "A New Fast Fuzzy Color Transfer Method",
%In the 4th IEEE International Symposium on Signal Processing and Information
%Technology (ISSPIT 2004), Rome, Italy, December 18–21, 2004.
%
%Usage:[R,F,t]=recolor(Is,Ir,Rs,Rr,Os,Or,W,a,b);
%Inputs:     Is:Source image.
%           Ir:Reference image.
%           Rs:Query regions in the source image.
%           Rr:Query regions in the reference image.
%           Os:Order of query regions in the reference image.
%           Or:Order of query regions in the source image.
%           W:Watermark signal.
%           a,b:Alpha and beta for the butterworth mapping function.
%Outputs:    R:Result.
%           F:Membership maps.
%           t:Elapsed time.
%Revision 1.0 Arash Abadpour Fall 2004 (Oct. 2, 2004)
```

Figure 3.82: Implementation of the *recolor* function.

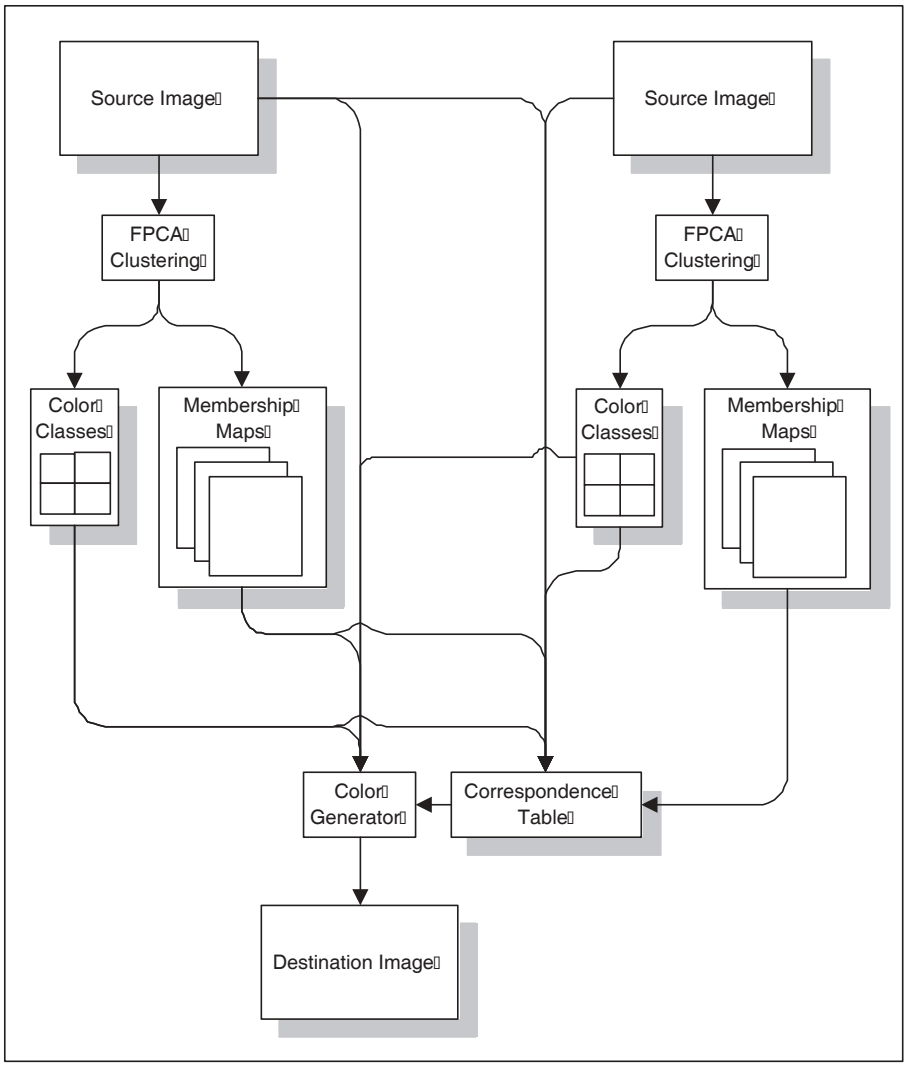


Figure 3.83: Flowchart of the proposed color transfer method.

source and the reference images to each of the c clusters, respectively. Note that the membership values of I_1 and I_2 pixels are measured regarding to the respective cluster parameters. Here, c is the number of the clusters which should be given to the FPCAC. In all experiments, we use three clusters, while it is observed that more number of clusters only inflates the processing time with no explicit influence on the quality of the results. The FPCAC also gives a color descriptor for each cluster in the source and the reference images. Denote them by $[\vec{\eta}_{11}, V_{11}], \cdots, [\vec{\eta}_{1c}, V_{1c}]$ and $[\vec{\eta}_{21}, V_{21}], \cdots, [\vec{\eta}_{2c}, V_{2c}]$, respectively. Here, $\vec{\eta}_{ij}$ is the fuzzy expectation of the vectors in the j–th cluster and V_{ij} denote the FPCA matrix of the j–th cluster.

Assume producing the crisp segmentation of the source image as the indexed image J_1. each pixel in J_1 holds an index in the range of $[1, \cdots, c]$ showing the cluster to which the corresponding pixel in the source image belongs more,

$$\forall x, y, J_1(x, y) = i \leftrightarrow \forall j \neq i, J_j(x, y) \leq J_i(x, y). \tag{3.9.261}$$

The crisp segmentation mask J_2 is produced in the same way for the reference image. Now, using all the pixels in the reference image that belong to the i–th cluster (decided by $J_1(x, y) = i$), we produce the image P_{i1}. In fact, the set of artificial images $[P_{11}, \cdots, P_{1c}]$ make a pallet for the source image. In the same way the pallet $[P_{21}, \cdots, P_{2c}]$ is produced for the reference image. Now the problem is to give an assignment between the corresponding pallet members in the source and the reference images. Although, the kind of heuristic assignments that *Greenfield et al.* [156] use

in their work is applicable here, we prefer to leave this part to the user. In this way, the method incorporates the intention of the user and gives better results. It should be emphasized that when no user intervention is desirable, the pallet members could be assigned according to their respective occupied area or any other criterion. Here, we show the pallet members to the user and ask for the index array m. In this way, m_i shows that the pallet representative P_{1i} in the source image is assigned to P_{2m_i} in the destination image.

Now, for the color vector $\vec{I}_{1xy}$ in the source image, its new version $\vec{I}'_{1xy}$ is computed as,

$$\vec{I}'_1(x,y) = \frac{\sum_{i=1}^{c} \tilde{J}_{1ixy}\Big(C_{2m_i}C_{1i}^{-1}(\vec{I}_{1xy} - \vec{\eta}_{1i}) + \vec{\eta}_{2m_i}\Big)}{\sum_{i=1}^{c} \tilde{J}_{1ixy}}, \tag{3.9.262}$$

where, $\tilde{J}_{1ixy}$ is the averaged version of J_{1ixy} using a convolution kernel of radius ρ. The reason for selecting $\tilde{J}_{1ixy}$ over the original J_{1ixy} is to add more blending to the vectors to prevent spurious edges. In this thesis, everywhere we use $\rho = 5$. Investigating (3.9.262) shows that the new vector $\vec{I}'_1(x,y)$ is produced using a weighted sum of the altered versions of the original vector $\vec{I}_1(x,y)$ using the parameters of each category. The weights of the sum are coming from the membership maps.

3.10 Image Registration

Let images I_1 and I_2 correspond to two different captures of the same scene in different times. The aim of the registration stage is to find the transform $T : [x,y] \rightarrow [x^\star, y^\star]$

in the way that, when applying the transform T on the image I_2 the resulting image, I_2', gets aligned on the image I_1. Here, we use some control points in the two images. We call the control points in the two images of I_1 and I_2 as $\vec{x}_i$ and $\vec{y}_i$ for $i = 1 \cdots n$, respectively. They are chosen so that applying the transform T on $\vec{x}_i$, the result lies on $\vec{y}_i$. In fact, $\vec{x}_i$ and $\vec{y}_i$ correspond to the same physical point in the imaged area. Here, we assume that the control points are given by human intervention. Also, we assume that they are properly distributed all over the images (we will investigate the effects of non–distributed control points). The proposed registration method uses numerical enhancements (Section 3.10.3, Section 3.10.3) of an approximate geometrical estimation (Section 3.10.2). The set of functions *CenterFP*, *ErrorFP*, *FindRegisteration*, *FindRegisterationNewton*, *FindRotation*, *FindScaling*, *FindScalinginternal*, *FindTranslation*, *FPHandler.m getFP*, *Hessiant*, *Jacobiant*, *merget.m ReadFP*, *register*, *registerfp*, *RegisterMerge*, *SaveFP*, and *showFP* perform the method described here. Note that the approach described here is only backed by a mathematical motivation to solve the registration problem geometrically. In this simple framework, using an appropriate mean square difference will yield the result.

3.10.1 Direct Linear Transformation and Affine Transform

Registration has an structural relation to the problem of camera calibration [93], where one is concerned with estimating the 3–D coordinates of a point from its corresponding 2–D coordinates in (at least) two different cameras. A well–known

CenterFP

```
function index=CenterFP(X,Y)
%Function center fixed points finds the coordinates of teh central fixed
%point.
%Usage:      index=CenterFP(X,Y);
%Inputs:     X,Y:Fixed points.
%Outputs:    index:The central fixed point.
%
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.84: Implementation of the *CenterFP* function.

ErrorFP

```
function r=ErrorFP(X1,Y1,X2,Y2,I)
%Function error fixed points computes the error!
%Usage:      r=ErrorFP(X1,Y1,X2,Y2,I);
%Inputs:     X,Y:Fixed points.
%                  I:Reference image.
%Outputs:    r:The error.
%
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.85: Implementation of the *ErrorFP* function.

FindRegisteration

```
function [Tx,Ty,er]=FindRegisteration(X1,Y1,X2,Y2,I1)
%Function find registeration finds the registeration of 2 on 1.
%Usage:      [Tx,Ty,er]=FindRegisteration(X1,Y1,X2,Y2,I1);
%Inputs:     X1,Y1:Fixed points on the original image.
%                 X2,Y2:Fixed points on the distorted image.
%                 I1:The original image, only used for finding the sizes.
%Outputs:    Tx,Ty:The transformation pair.
%                 er:The final error.
%
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.86: Implementation of the *FindRegisteration* function.

FindRegisterationNewton

```
function [Tx,Ty,er]=FindRegisterationNewtonRaphson(X1,Y1,X2,Y2,I1,res)
%Function find registeration Newton Raphson finds the registeration of 2 on 1.
%Usage:      [Tx,Ty,er]=FindRegisterationNewtonRaphson(X1,Y1,X2,Y2,I1,res);
%Inputs:     X1,Y1:Fixed points on the original image.
%                 X2,Y2:Fixed points on the distorted image.
%                 I1:The original image, only used for finding the sizes.
%                 Step:Step.
%                 res:Error resolution.
%Outputs:    Tx,Ty:The transformation pair.
%                 er:The error.
%
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.87: Implementation of the *FindRegisterationNewton* function.

FindRotation

```
function [tx,ty,er]=FindRotation(X1,Y1,X2,Y2)
%Function find rotation, finds the rotation to best register 2 on 1.
%Usage:      [tx,ty]=FindRotation(X1,Y1,X2,Y2)
%Inputs:      X1,Y1:Fixed points in the source image.
%                  X2,Y2:Fixed points in the distorted image.
%Outputs:   tx,ty:The transformations.
%                  er:Error;
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.88: Implementation of the *FindRotation* function.

FindScaling

```
function [tx,ty,er]=FindScaling(X1,Y1,X2,Y2,thr)
%Function find scaling, finds the scaling that registeres 2 on 1
%Usage:      [tx,ty,er,doagain]=FindScaling(X1,Y1,X2,Y2,thr)
%Inputs:      X1,Y1:Fixed points in the source image.
%                  X2,Y2:Fixed points in the distorted image.
%                  thr:Marginal unbalance between horizental and vertical
%                  scales.
%Outputs:   tx,ty:The transformations.
%                   er:The error.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.89: Implementation of the *FindScaling* function.

FindScalinginternal

```
function [tx,ty,er,doagain]=FindScalingInternal(X1,Y1,X2,Y2,thr)
%Function find scaling, is an internal function used in the find scaling
%function.
%Usage:      [tx,ty,er,doagain]=FindScalingInternal(X1,Y1,X2,Y2,thr)
%Inputs:      X1,Y1:Fixed points in the source image.
%                  X2,Y2:Fixed points in the distorted image.
%                  thr:Marginal unbalance between horizental and vertical
%                  scales.
%Outputs:   tx,ty:The transformations.
%                   er:The error.
%                  doagain:Do the function again.
%Never use this function manually if you are not sure what it does!
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.90: Implementation of the *FindScalinginternal* function.

FindTranslation

```
function [tx,ty,er]=FindTranslation(X1,Y1,X2,Y2)
%Function find translation, finds the translation that registeres 2 on 1
%Usage:      [tx,ty,er]=FindTranslation(X1,Y1,X2,Y2)
%Inputs:      X1,Y1:Fixed points in the source image.
%                  X2,Y2:Fixed points in the distorted image.
%Outputs:   tx,ty:The transformations.
%                   er:The error.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.91: Implementation of the *FindTranslation* function.

merget

```
function [Tx,Ty]=merget(tx1,ty1,tx2,ty2)
%Function merge transformation finds the t2ot1.
%Usage:      [Tx,Ty]=merget(tx1,ty1,tx2,ty2)
%Inputs:     tx1,ty1:The first transformation.
%                 tx1,ty1:The second transformation.
%Outputs:    Tx,Ty:The resulting transformation.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.92: Implementation of the *merget* function.

getFP

```
function [X,Y]=GetFP(I,n,x,y)
%Function get fixed points gets the coordinates of n points from the user.
%Usage:      [X,Y]=GetFP(I,n,x,y);
%Inputs:     I:The image.
%            n:n>0: Numebr of the points.
%              n<0:Edit the location of the -n-th point.
%            x,y:The points. Only necessary when n<0.
%Outputs:    X,Y:Fixed points.
%
%Revision 1.1 Arash Abadpour Summer 2004 (1383/4/8)
```

Figure 3.93: Implementation of the *getFP* function.

ReadFP

```
function [X,Y]=ReadFP(index)
%Function read fixed points reads the information in a mat file and gives the
   X and Y information.
%Usage:      [X,Y]=ReadFP(index);
%Inputs:     index:The data will be loaded from index.mat.
%Outputs:    X,Y: Fixed points.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.94: Implementation of the *ReadFP* function.

Hessiant

```
function H=Hessiant(X,Y,t)
%Function transform Hessian computes the hessian matrix of the transform t for
%the points given by X and Y.
%Usage:      H=Hessiant(X,Y,t);
%Inputs:     X,Y:Input points.
%           t:The transform.
%Outputs:    H:The hessian matrix.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.95: Implementation of the *Hessiant* function.

Jacobiant

```
function J=Jacobiant(X,Y,u,t)
%Function trasnform Jacobian computes the Jacobian vector of the transform t for
%the points given by X and Y.
%Usage:      J=Jacobiant(X,Y,u,t);
%Inputs:     X,Y:Input points.
%           Z:The output;
%           t:The transform.
%Outputs:    J:The Jacobian vector.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.96: Implementation of the *Jacobiant* function.

register

```
function [J1,J2]=register(I1,I2,tx,ty,cut)
%Function register registers the two images I1 and I2 into a single image.
%Usage:      [J1,J2]=register(I1,I2,tx,ty,cut);
%Inputs:     I1:The original image.
%           I2:The image to be registered.
%           tx,ty:The transformations.
%           cut:Cut the result to the first image bounderies?
%Outputs:    J1,J2:The resulting images.
%Revision 1.001 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.97: Implementation of the *register* function.

registerfp

```
function [X,Y]=registerfp(x,y,tx,ty)
%Function register fixed points applies the transformation  on the fixed
%points.
%Usage:      [X,Y]=registerfp(x,y,tx,ty)
%Inputs:     x,y:The fixed points.
%                  tx,ty:The transformations.
%Outputs:    X,Y:The result.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.98: Implementation of the *registerfp* function.

RegisterMerge

```
function I=RegisterMerge(I1,I2,tx,ty,cut)
%Function register merge registers the two images I1 and I2 into a single image.
%Usage:      I=RegisterMerge(I1,I2,tx,ty,cut)
%Inputs:     I1:The original image.
%                  I2:The image to be registered.
%                  tx,ty:The transformations.
%                cut:Cut the result to the first image bounderies?
%Outputs:    I:The resulting image.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.99: Implementation of the *RegisterMerge* function.

SaveFP

```
function SaveFP(X,Y,index)
%Function save fixed points saves the information in X and Y in a mat file.
%Usage:      SaveFP(X,Y,index)
%Inputs:     X,Y:Input points.
%           index:The data will be saved in index.mat.
%Outputs:
%Revision 1.0 Arash Abadpour Summer 2004 (1383/4/7)
```

Figure 3.100: Implementation of the *SaveFP* function.

showFP

```
function fig=showFP(X,Y,I,showlines)
%Function show fixed points visualizes the image I and the fixed points
%listed in X and Y.
%Usage:      fig=showFP(X,Y,I,showlines)
%Inputs:     X,Y:Input points.
%                I:The image.
%                showlines:Show the lines betweenthe central point and
%                the fized points.
%Outputs:    f:Figure handle.
%Revision 1.1 Arash Abadpour Summer 2004 (1383/4/8)
```

Figure 3.101: Implementation of the *showFP* function.

model for camera projection is the *direct linear transform* (DLT) by *Abdel–Aziz* and *Karara* [46]. Modelling a camera with 11 parameters, DLT is able to compensate perspective distortions [46].

In the methodology of DLT, each camera is modelled by 11 parameters and the projection of the point $\vec{X} = [x, y, z]$ on a camera is defined as [46],

$$u = \frac{a_u x + b_u y + c_u z + d_u}{ax + by + cz + 1}, \tag{3.10.263}$$

$$v = \frac{a_v x + b_v y + c_v z + d_v}{ax + by + cz + 1}. \tag{3.10.264}$$

Here, the denominator term,

$$\lambda = ax + by + cz + 1, \tag{3.10.265}$$

applies the effects of the destination from $\vec{X}$ to the center of the camera on the projected point coordinates [46]. In the case of space–born imagery, there are two simplifications to be applied on the DLT formulation. Firstly, the vertical distance between the camera and the subject points, z, is constant (because the camera plane is almost parallel to the subject [414]). Secondly, as the normal vector of the camera plane and the normal vector of the earth surface are almost parallel, the denominator term, λ, gets constant for all image pixels. Thus, setting,

$$a = \frac{1}{\lambda}a_u, b = \frac{1}{\lambda}b_u, c = \frac{1}{\lambda}(c_u z + d), \tilde{a} = \frac{1}{\lambda}a_v, \tilde{b} = \frac{1}{\lambda}b_v, \tilde{c} = \frac{1}{\lambda}(c_v z + d), \tag{3.10.266}$$

gives the simplified linear model of,

$$u = ax + by + c, \tag{3.10.267}$$

Table 3.4: Some common transforms in pair notation.

Description	**Transformation**	
	t_x	t_y
Unity.	$[1, 0, 0]$	$[0, 1, 0]$
Translation by $[x_0, y_0]^T$.	$[1, 0, x_0]$	$[0, 1, y_0]$
θ radians rotation around $[0, 0]$.	$[\cos\theta, -\sin\theta, 0]$	$[\sin\theta, \cos\theta, 0]$
Scaling by α (balanced).	$[\alpha, 0, 0]$	$[0, \alpha, 0]$
Scaling by α horizontally and by β vertically (unbalanced).	$[\alpha, 0, 0]$	$[0, \beta, 0]$

$$v = \tilde{a}x + \tilde{b}y + \tilde{c}, \tag{3.10.268}$$

also known as the affine transform [414]. The affine transform can be written in the matrix notation as,

$$\vec{u} = \begin{pmatrix} a & b \\ \tilde{a} & \tilde{b} \end{pmatrix} \vec{x} + \begin{pmatrix} c \\ \tilde{c} \end{pmatrix}. \tag{3.10.269}$$

Also, the two different transformations in the affine transform resulting in u and v, are independently parameterized, in contrast with the mixed case in the conventional DLT. While in proving the theorems we use the matrix notation, in the text we show the transformation described in the affine transformation as the pair of,

$$T = (t_x, t_y) = ([a, b, c], [\tilde{a}, \tilde{b}, \tilde{c}]). \tag{3.10.270}$$

Table 3.4 lists some common transforms in this notation.

For two pairs of transforms, (t_x, t_y) and (t'_x, t'_y), we define the transform (t''_x, t''_y) as the result of merging (t_x, t_y) and (t'_x, t'_y), writing as,

$$(t''_x, t''_y) = (t'_x, t'_y) \circ (t_x, t_y), \tag{3.10.271}$$

when applying (t''_x, t''_y) on an image results the same as applying (t_x, t_y) on the original image and then applying (t'_x, t'_y) on the result. Note that in the general case $(t'_x, t'_y) \circ (t_x, t_y)$ and $(t_x, t_y) \circ (t'_x, t'_y)$ are not necessarily equivalent. Numerical derivation shows that using the transform pair notation, $(t''_x, t''_y) = (t'_x, t'_y) \circ (t_x, t_y)$ implies that,

$$a'' = a'a + b'\tilde{a}, b'' = a'b + b'\tilde{b}, c'' = a'c + b'\tilde{c} + c', \tag{3.10.272}$$

$$\tilde{a}'' = \tilde{a}'a + \tilde{b}'\tilde{a}, \tilde{b}'' = \tilde{a}'b + \tilde{b}'\tilde{b}, \tilde{c}'' = \tilde{a}'c + \tilde{b}'\tilde{c} + \tilde{c}'. \tag{3.10.273}$$

Note that applying numerous transformation pairs on an image is a lengthy operation, also resulting in numerical errors. The above merging scheme enables us to merge the transformation pairs and produce just one transformation which is applied much faster and more reliably.

3.10.2 Geometrical Estimation

The proposed method for estimating the transformation registering two given remotely sensed images, uses it decomposition into three elementary transformations of translation, rotation and un–balanced scaling (see Figure 3.102). In practice, the elementary transformations are also decomposed into other transformations and then the result of merging the combination of all transforms is applied on the image.

Scaling

As shown in Table 3.4, a balanced scaling is an orthogonal transformation which multiples all distances by a constant factor, while preserving angles. In contrast, an

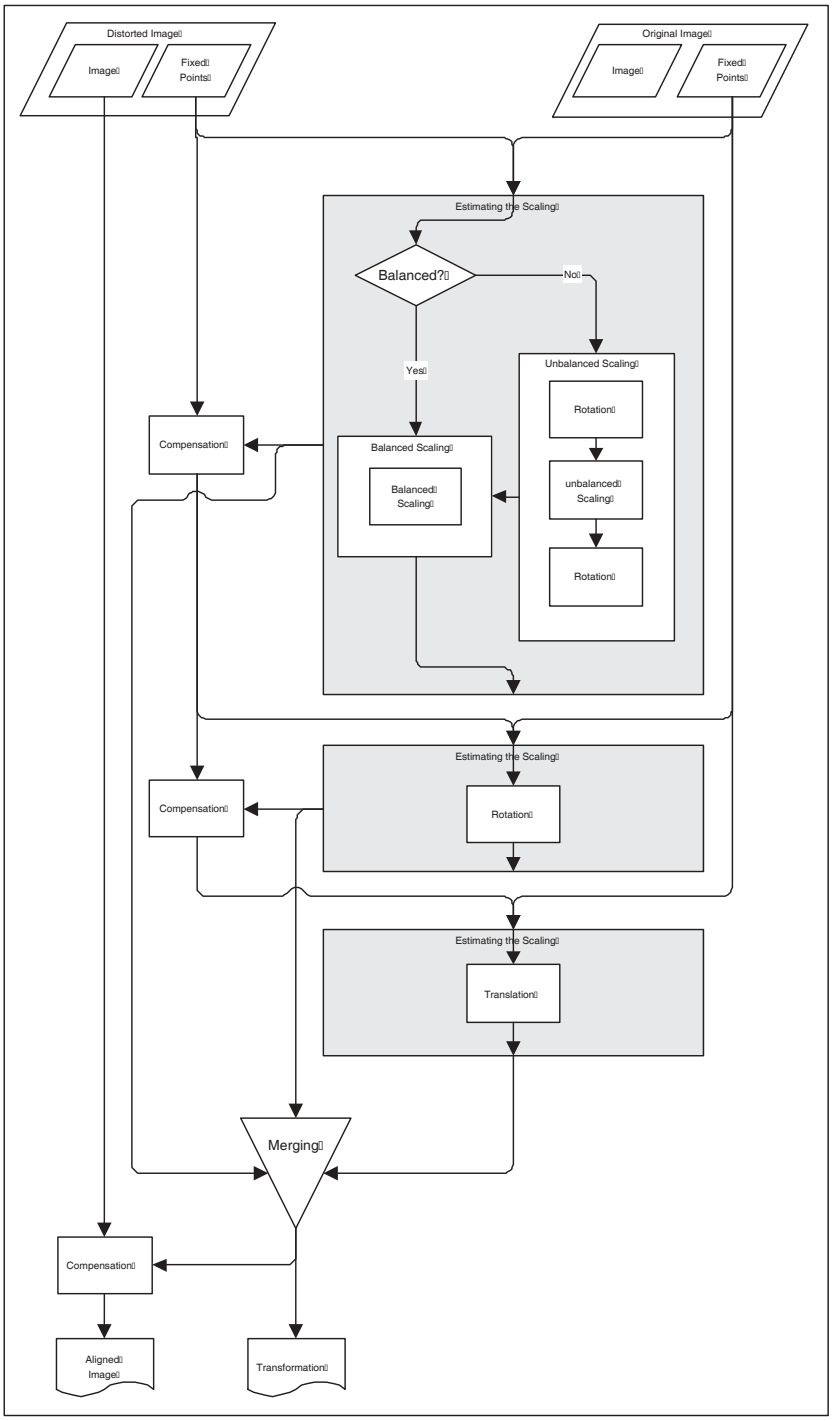


Figure 3.102: Flowchart of the proposed registration method.

unbalanced transformation manipulates both angles and distances in a more complex way. It is known from linear algebra that, for an unbalanced scaling, there exists two perpendicular vectors $\vec{v}_1$ and $\vec{v}_2$ which are magnified by factors of α_1 and α_2 and the result of the transformation on any arbitrary vector can be measured using these two vectors. For an arbitrary vector $\vec{x}$ there is one and only one representation as,

$$\vec{x} = x_1\vec{v}_1 + x_2\vec{v}_2. \tag{3.10.274}$$

As the scaling transformation ($\vec{y} = S\vec{x}$) is a linear operator we have,

$$S\vec{x} = x_1\alpha_1\vec{v}_1 + x_2\alpha_2\vec{v}_2. \tag{3.10.275}$$

Assuming the points $\vec{x}_i$ on the unit circle, we want to find the locus of the transformed points $\vec{y}_i = S\vec{x}_i$. Assume that by changing the coordinate system to the one with the axis of $\vec{v}_1$ and $\vec{v}_2$ the point $\vec{x}_i$ in the original coordinate system changes to $\vec{x}'_i$. Now, the points $\vec{x}'_i$ ar also on the unit circle (changing the coordinate system is just a rotation).By writing the transformation matrix S as its diagonal decomposition form of VDV^{-1}, the matrix V contains the two vectors $\vec{v}_1$ and $\vec{v}_2$ as its columns and D is a diagonal matrix with terms α_1 and α_2 on its main diagonal,

$$S = \begin{bmatrix} v_{1x} & v_{2x} \\ v_{1y} & v_{2y} \end{bmatrix} \begin{bmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{bmatrix} \begin{bmatrix} v_{1x} & v_{1y} \\ v_{2x} & v_{2y} \end{bmatrix}.$$

Let $\vec{z}_i = V^{-1}\vec{y}_i$. $\vec{z}_i$ denotes the representation of $\vec{y}_i$ in the new coordinate system. Now, we have,

$$\vec{z}_i^T D^{-2}\vec{z}_i = \vec{y}_i^T V D^{-2} V^{-1}\vec{y}_i = \vec{x}_i^T S' V D^{-2} V^{-1} S\vec{x}_i, \tag{3.10.277}$$

$$\vec{z}_i^T D^{-2} \vec{z}_i = \vec{x}_i^T V D V^{-1} V D^{-2} V^{-1} V D V^{-1} \vec{x}_i = \vec{x}_i^T \vec{x}_i = \|\vec{x}_i\|^2 = 1. \tag{3.10.278}$$

Thus, $\vec{y}_i$s form an ellipse rotated by V with diagonal equal to $2\alpha_1$ and $2\alpha_2$.

Consider $\vec{y} = S\vec{x}$ again with the constraint,

$$\|\vec{x}\| = 1. \tag{3.10.279}$$

Using the coordinates of $\vec{x}$ in the $\{\vec{v}_1, \vec{v}_2\}$ system (see (3.10.274)) and considering the orthogonality condition ($\vec{v}_1 \perp \vec{v}_2$) we have,

$$x_1^2 + x_2^2 = 1, \|\vec{y}\|^2 = a^2\alpha_1^2 + b^2\alpha_2^2. \tag{3.10.280}$$

Substituting,

$$x_2^2 = 1 - x_1^2, \tag{3.10.281}$$

into,

$$\Delta = x_1^2\alpha_1^2 + x_2^2\alpha_2^2, \tag{3.10.282}$$

and computing $\frac{\partial \Delta}{\partial a^2}$ to find the extremes of Δ, yields the values of α_1^2 and α_2^2 for the vectors $\vec{x} = \vec{v}_1$ and $\vec{x} = \vec{v}_2$, respectively. Thus, for the control points $\vec{y}_i$ in the second image corresponding to the control points $\vec{x}_i$ located on the unit circle in the first image, and performing the PCA gives the matrix V. The above mentioned method is only applicable if we have the set of control points making a unit circle in the first image.

The geometric model that we use is,

$$\vec{y} = R(VDV^{-1}(\vec{x} - \vec{a}) - \vec{b}) - \vec{c}, \tag{3.10.283}$$

where, R is the rotation matrix and VDV^{-1} is the unbalanced scaling matrix. Here the two matrices R and V are orthonormal, D is diagonal, the vectors $\vec{a}$ and $\vec{b}$ model the points around which the scaling and rotation take place, and $\vec{c}$ is the final translation vector. Equation (3.10.283) can be rewritten in the simpler form of,

$$\vec{y} = A\vec{x} + \vec{B}, \tag{3.10.284}$$

where,

$$A = RVDV^{-1}. \tag{3.10.285}$$

$$\vec{B} = -(RVDV^{-1}\vec{a} + R\vec{b} + \vec{c}). \tag{3.10.286}$$

Consider the vectors $\vec{x}_i = \vec{x}_0 + \vec{v}_i$ in the first image, where $\vec{v}_i$s satisfy $\|\vec{v}_i\| = 1$ and $\vec{y}_i$ is the vector corresponding to $\vec{x}_i$ in the second image. Then, $\vec{y}_i - \vec{y}_0$ is the result of applying the transformation on $\vec{x}_i - \vec{x}_0$ (the transformation is linear). Assume the case of $\vec{v}_2 = \alpha\vec{v}_1$ and the two vectors $\vec{y}_1$ and $\vec{y}_2$ as the result of the transformation shown in (3.10.283) applied on the two vectors $\vec{x}_1 = \vec{x}_0 + \vec{v}_1$ and $\vec{x}_2 = \vec{x}_0 + \vec{v}_2$, respectively. We have,

$$\frac{\|\vec{y}_2 - \vec{y}_0\|}{\|\vec{y}_1 - \vec{y}_0\|} = \frac{\|A\vec{v}_2\|}{\|A\vec{v}_1\|} = \alpha. \tag{3.10.287}$$

Thus, for the arbitrary vector $\vec{x}_0$ in the first image, the length of the vector $\vec{x} - \vec{x}_0$ in the first image and the length of its image in the second image just vary by a constant multiplier, which is the scaling factor in that direction. Hence, for the set of n control points $\vec{x}_i$ and $\vec{y}_i$ in the first image and the second image, the principal directions of

the set,

$$\Phi = \left\{ \frac{\vec{y}_i - \vec{y}_{i_0}}{\left\| \vec{x}_i - \vec{x}_{i_0} \right\|} \middle| i = 1, \cdots, n, i \neq i_0, 1 \leq i_0 \leq n \right\}, \tag{3.10.288}$$

gives the matrix V, while its eigenvalues are proportional to α_1 and α_2. By using the computed V and D, the unbalanced scaling is compensated while one needs to compensate the balanced scaling independently. Note that if the scaling is balanced, Φ makes a circle and the eigenvectors get ambiguous. We propose to compute the standard deviation of the set

$$\tilde{\Phi} = \left\{ \frac{\left\| \vec{y}_i - \vec{y}_{i_0} \right\|}{\left\| \vec{x}_i - \vec{x}_{i_0} \right\|} \middle| i = 1, \cdots, n, i \neq i_0, 1 \leq i_0 \leq n \right\}, \tag{3.10.289}$$

as the criteria for the scale being balanced. Note that $\sigma_{\tilde{\Phi}} \simeq 0$ show that the scaling is balanced. Also, note that for the balanced scaling, $\eta_{\tilde{\Phi}}$ gives the scale. To find a value of i_0 which results in computing the scales in more directions, we propose to use the point in the control points set, which is the nearest to the expectation vector of the control points. Note that selecting any other member of the control points set as the central point yields the same result. After estimating the scaling, we apply the produced transformation pair on the $\vec{y}_i$ to form the new control points in the second image. For simplicity we use the same $\vec{y}_i$ term for the new points. Figure 3.103 shows the pseudo–code of the proposed algorithm for estimating the scaling.

- ***Aim***: *Estimating the balanced and unbalanced scaling.*
- ***Inputs***:
  - *The set of corresponding control points* $\vec{x}_i$ *and* $\vec{y}_i$ *in the first image and the second image, respectively.*
  - *The balanced threshold* ε_s.
- ***Output***:
  - *A pair of transformations* T.
- ***Method***:

  1 *Select the central point* i_0.

  2 *Compute* Φ, $\sigma_{\tilde{\Phi}}$, *and* $\eta_{\tilde{\Phi}}$.

  3 *if* $\sigma_{\tilde{\Phi}} < \varepsilon_s$ *then go to the balanced scaling (4), else goto the unbalanced scaling (5).*

  4 *(Balanced Scaling) Return* $T = ([\eta_{\tilde{\Phi}}^{-1}, 0, 0], [0, \eta_{\tilde{\Phi}}^{-1}, 0])$.

  5 *(Unbalanced Scaling) Perform the PCA on* Φ *and compute* $\vec{v}_1$ *and* $\vec{v}_2$ *as the eigenvectors and* α_1 *and* α_2 *as the eigenvalues.*

  6 $t_1 = ([v_{1x}, v_{1y}, 0], [v_{2x}, v_{2y}, 0])$.

  7 $t_2 = ([\alpha_1^{-1}, 0, 0], [0, \alpha_2^{-1}, 0])$.

  8 $t_3 = ([v_{1x}, v_{2x}, 0], [v_{1y}, v_{2y}, 0])$.

  9 $t_{ub} = t_3 \circ t_2 \circ t_1$.

  10 $\vec{y}_i = t_{ub}(\vec{y}_i)$.

  11 *Compute* Φ *and* $\eta_{\tilde{\Phi}}$ *again.*

  12 $t_4 = ([\eta_{\tilde{\Phi}}^{-1}, 0, 0], [0, \eta_{\tilde{\Phi}}^{-1}, 0])$.

  13 *Return* $T = t_4 \circ t_{ub}$.

Figure 3.103: Pseudo–code of the proposed algorithm for estimating the scaling.

Rotation

Now, the transformation is reduced to,

$$\vec{y} = R\vec{x} + \vec{b}, \tag{3.10.290}$$

where R is a rotation matrix. Thus, for an arbitrary value of i_0 and $i \neq i_0$ we have,

$$\vec{y}_i - \vec{y}_{i_0} = R(\vec{x}_i - \vec{x}_{i_0}). \tag{3.10.291}$$

The actual angle between two vectors in the 2–D plane is computed as,

$$\angle(\vec{v}, \vec{u}) = sgn\Big((\vec{v} \times \vec{u}) \cdot \vec{j}\Big) \cos^{-1} \frac{\vec{v} \cdot \vec{u}}{\|\vec{v}\|\|\vec{u}\|}, \tag{3.10.292}$$

where $sgn(x)$ is the *signum* function, defined as:

$$sgn(x) = \begin{cases} 1 \ , x > 0 \\ 0 \ , x = 0 \\ -1 \ , x < 0 \end{cases} . \tag{3.10.293}$$

Now, by forming the set

$$\Theta = \left\{ \angle\Big(\vec{y}_i - \vec{y}_{i_0}, \vec{x}_i - \vec{x}_{i_0}\Big) \middle| i = 1, \cdots, n, i \neq i_0, 1 \leq i_0 \leq n \right\}, \tag{3.10.294}$$

and computing,

$$\eta_\Theta = E\Big\{\theta | \theta \in \Theta\Big\}, \tag{3.10.295}$$

the rotation angle is estimated. After estimating the rotation, we apply the produced transformation pair on $\vec{y}_i$ to form new vectors. For simplicity we continually call the new control points as $\vec{y}_i$. Figure 3.104 shows the pseudo–code of the proposed algorithm for estimating the rotation.

- ***Aim***: *Estimating the rotation.*
- ***Inputs***:
  - *The set of corresponding control points $\vec{x}_i$ and $\vec{y}_i$ in the first image and the second image, respectively.*
- ***Output***:
  - *A pair of transformations T.*
- ***Method***:

  1 Select the central point i_0.

  2 Compute Θ and η_Θ.

  3 Return $T = ([\cos\eta_\Theta, \sin\eta_\Theta, 0], [-\sin\eta_\Theta, \cos\eta_\Theta, 0])$.

Figure 3.104: Pseudo–code of the proposed algorithm for estimating the rotation.

Translation

Now, the transformation is reduced to a translation,

$$\vec{y} = \vec{x} + \vec{a}. \tag{3.10.296}$$

Note that,

$$\vec{a} = E\{\vec{y}_i\} - E\{\vec{x}_i\}, \tag{3.10.297}$$

gives the translation vector. Figure 3.105 shows the pseudo–code of the proposed algorithm for estimating the translation.

- ***Aim***: *Estimating the translation.*
- ***Inputs***:
  - *The set of corresponding control points $\vec{x}_i$ and $\vec{y}_i$ in the first image and the second image, respectively.*
- ***Output***:
  - *A pair of transformations T.*
- ***Method***:

  1 Select the central point i_0.
  2 Compute $\vec{\eta} = E\{\vec{y}_i\}$ and $\vec{\rho} = E\{\vec{x}_i\}$.
  3 Return $T = ([1, 0, -\eta_x + \rho_x], [0, 1, -\eta_y + \rho_y])$.

Figure 3.105: Pseudo–code of the proposed algorithm for estimating the translation.

Finalizing

Now merging the results, the whole transformation needed to register the second image on the first image is computed.

It is useful to show the registration result in a visual way for illustration purposes. We propose to use a color image when the underlying images are gray–scales. By putting the first image in the red plane and the registered version of the second image in the green plane while the blue plane is filled with 255, one can find the non–alignments pixels which will be shown as magenta and cyan regions.

As a numerical criteria of matching, we propose to use the normalized mean square

error of the control points as the total distortion of,

$$\varepsilon^2 = \frac{1}{n(w^2+h^2)} \sum_{i=1}^{n} \|\vec{y}_i - \vec{x}_i\|^2, \tag{3.10.298}$$

where, the first image size is $h \times w$ (in pixels). We call this method the geometric estimation method.

3.10.3 Numerical Enhancement

In this section we examine two numerical methods of gradient–descent and Newton optimization to enhance the results of the geometric estimation method proposed in Section 3.10.2. Both methods are based on a novel formulation of the transformation function error.

Gradient–Descent Enhancement

Assume the function $f : R^n \rightarrow R$. The gradient–descent method, is a first–order method for finding the minimum point of this function, starting from a good initialization. The standard gradient descent method uses,

$$\vec{x}^{\star} = \vec{x} + \lambda \vec{\nabla} f, \tag{3.10.299}$$

as the next step, where λ is a user defined parameter, (usually smaller than unity). We use a more controlled formula here, defined as:

$$\vec{x}^{\star} = \begin{cases} \vec{x} + \frac{\lambda}{\|\vec{\nabla} f\|} \vec{\nabla} f \ , \|\vec{\nabla} f\| > \lambda \\ \vec{x} + \vec{\nabla} f \ , \|\vec{\nabla} f\| \leq \lambda \end{cases} , \tag{3.10.300}$$

where, λ is the user defined step size. This method adds more inertia to the solution.

Assume the first image I_1 on which the second image I_2 is to be registered. Also, assume that the method proposed in Section 3.10.2 has resulted in the transformation T with total distortion of ε. Note that the transform T produces the point (u, v) out of (x, y) as,

$$\begin{cases} u = \alpha x + \beta y + \gamma \\ v = \alpha' x + \beta' y + \gamma' \end{cases} . \tag{3.10.301}$$

Lets rewrite (3.10.301) as,

$$\begin{cases} u = a\alpha x + b\beta y + c\gamma \\ v = a'\alpha' x + b'\beta' y + c'\gamma' \end{cases} , \tag{3.10.302}$$

where, a, b, c, a', b', and c' are all equal to unity. By tuning these parameters around 1, one can enhance the results. Now, defining,

$$\Psi(\vec{\omega}) = \sum_{i=1}^{n} \Big[(x_{ix} - a\alpha y_{ix} - b\beta y_{iy} - c\gamma)^2 + (x_{iy} - a'\alpha' y_{ix} - b'\beta' y_{iy} - c'\gamma')^2 \Big], \tag{3.10.303}$$

where $\vec{x}_i = [x_{ix}, x_{iy}]$ and $\vec{y}_i = [y_{ix}, y_{iy}]$ are the i-th corresponding control points in the first and the second images, respectively and $\vec{\omega}$ is the vector $[a, b, c, a', b', c']$. We form the gradient vector of Ψ, as:

$$\vec{\nabla}\Psi(\vec{\omega})\Big|_{\vec{\omega}=\vec{1}} = -2 \sum_{i=1}^{n} \begin{pmatrix} \alpha d_{ix} y_{ix} \\ \beta d_{ix} y_{iy} \\ \gamma d_{ix} \\ \alpha' d_{iy} y_{ix} \\ \beta' d_{iy} y_{iy} \\ \gamma' d_{iy} \end{pmatrix}, \tag{3.10.304}$$

where,

$$d_{ix} = x_{ix} - \alpha y_{ix} - \beta y_{iy} - \gamma, \tag{3.10.305}$$

$$d_{iy} = x_{iy} - \alpha' y_{ix} - \beta' y_{iy} - \gamma'. \tag{3.10.306}$$

We propose to apply (3.10.302) on (3.10.304) to minimize (3.10.303) starting from T, iteratively. Also, we propose the halting condition defined as ε being smaller than the preselected margin of $\tilde{\varepsilon}$ or the number of iterations becoming too much.

Thus, in each iteration the transformation,

$$T = (t_x, t_y), t_x = [\alpha, \beta, \gamma], t_y = [\alpha', \beta', \gamma'], \tag{3.10.307}$$

changes to,

$$T^\star = (t_x^\star, t_y^\star), \tag{3.10.308}$$

$$t_x^\star = [\alpha(1+\delta_\alpha), \beta(1+\delta_\beta), \gamma(1+\delta_\gamma)], \tag{3.10.309}$$

$$t_y^\star = [\alpha'(1+\delta_{\alpha'}), \beta'(1+\delta_{\beta'}), \gamma'(1+\delta_{\gamma'})], \tag{3.10.310}$$

where,

$$[\delta_\alpha, \delta_\beta, \delta_\gamma, \delta_{\alpha'}, \delta_{\beta'}, \delta_{\gamma'}]^T = \begin{cases} \frac{\lambda}{\|\vec{\nabla}\Psi\|}\vec{\nabla}\Psi \ , \|\vec{\nabla}\Psi\| > \lambda \\ \vec{\nabla}\Psi \ , \|\vec{\nabla}\Psi\| \leq \lambda \end{cases} \Bigg|_{\vec{\omega}=\vec{1}}. \tag{3.10.311}$$

Although, gradient descent method is proved to go toward the minimum points of the function when λ is selected small enough, by reducing λ the convergence speed declines.

Newton Enhancement

Assume the function $f : R^n \rightarrow R$. The Newton method, is a second–order method for finding an extremum point of this function, starting from a good initialization. At the extremum point we have,

$$\frac{\partial f}{\partial x_i} = 0, i = 1 \cdots n, \tag{3.10.312}$$

or equivalently,

$$\vec{\nabla} f = \vec{0}. \tag{3.10.313}$$

Defining,

$$\vec{g}(\vec{x}) = \vec{\nabla} f|_{\vec{x}}, \tag{3.10.314}$$

produces the new function $g : R^n \rightarrow R^n$ getting its zero vector at the extremum points of f. Using the Tailor series expansion of $\vec{g_i}$ only to the first derivative, results in,

$$g_i(\vec{x}) = g_i(\vec{x}_0) + \vec{\nabla} g_i^T|_{\vec{x}_0}(\vec{x} - \vec{x}_0), \tag{3.10.315}$$

where, $\vec{\nabla} g_i$ is the Gradient of g_i defined as,

$$\vec{\nabla} g_i = \begin{pmatrix} \frac{\partial g_i}{\partial x_1} \\ \vdots \\ \frac{\partial g_i}{\partial x_n} \end{pmatrix}. \tag{3.10.316}$$

Setting $g_i(\vec{x}) = 0$ in (3.10.315) results in,

$$\vec{\nabla} g_i^T|_{\vec{x}_0}(\vec{x} - \vec{x}_0) = -g_i(\vec{x}_0), \tag{3.10.317}$$

which is a linear function of $\vec{dx} = \vec{x} - \vec{x}_0$. Combining the n equations produced in this way for $i = 1 \cdots n$, results in,

$$H(f)(\vec{x}^\star - \vec{x}) = -g_i(\vec{x}), \tag{3.10.318}$$

or equivalently,

$$\vec{x}^\star = \vec{x} - H(f)^{-1} g_i(\vec{x}), \tag{3.10.319}$$

as the iteration of the Newton method towards the extremum. The matrix $H(f)$ is the Hessian matrix of the function f, defined as:

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ & \vdots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}. \tag{3.10.320}$$

Note that the Newton method finds the local extremum, not the local minimum. Thus, proper selection of the initial point is of great importance.

Assume the first image I_1 on which the second image I_2 is to be registered. Also assume that the method proposed in Section 3.10.2 has resulted in the transformation T with total distortion of ε. Here, we separate $\Psi(\vec{\omega})$ defined in (3.10.303) into two parts of

$$\Psi_x(\vec{\omega}_x) = \sum_{i=1}^{n} (x_{ix} - a\alpha y_{ix} - b\beta y_{iy} - c\gamma)^2, \vec{\omega}_x = [\alpha, \beta, \gamma], \tag{3.10.321}$$

$$\Psi_y(\vec{\omega}_y) = \sum_{i=1}^{n} (x_{iy} - a'\alpha' y_{ix} - b'\beta' y_{iy} - c'\gamma')^2, \vec{\omega}_y = [\alpha', \beta', \gamma']. \tag{3.10.322}$$

Hence,

$$\Phi(\vec{\omega}) = \Phi_x(\vec{\omega}_x) + \Phi_y(\vec{\omega}_y), \frac{\partial \Phi_x(\vec{\omega}_x)}{\vec{\omega}_y} = \vec{0}, \frac{\partial \Phi_y(\vec{\omega}_y)}{\vec{\omega}_x} = \vec{0}. \tag{3.10.323}$$

Thus, we propose to minimize Ψ_x and Ψ_y, independently. Numerical derivation shows that,

$$\vec{\nabla}\Psi_x(\vec{\omega}_x)|_{a,b,c=1} = -2\sum_{i=1}^{n}\begin{pmatrix}\alpha d_{ix}y_{ix}\\ \beta d_{ix}y_{iy}\\ \gamma d_{ix}\end{pmatrix},$$

$$H\Psi_x(\vec{\omega}_x)|_{a,b,c=1} = 2\sum_{i=1}^{n}\begin{pmatrix}\alpha^2 y_{ix}^2 & \alpha\beta y_{ix}y_{iy} & \alpha\gamma y_{ix}\\ \alpha\beta y_{ix}y_{iy} & \beta^2 y_{iy}^2 & \beta\gamma y_{iy}\\ \alpha\gamma y_{ix} & \beta\gamma y_{iy} & \gamma^2\end{pmatrix},$$

where,

$$d_{ix} = x_{ix} - \alpha y_{ix} - \beta y_{iy} - \gamma, \tag{3.10.326}$$

$$d_{iy} = x_{iy} - \alpha' y_{ix} - \beta' y_{iy} - \gamma'. \tag{3.10.327}$$

Thus, in each iteration, the transformation

$$T = (t_x, t_y), t_x = [\alpha, \beta, \gamma], t_y = [\alpha', \beta', \gamma'], \tag{3.10.328}$$

changes to,

$$T^\star = (t_x^\star, t_y^\star), \tag{3.10.329}$$

$$t_x^\star = [\alpha(1+\delta_\alpha), \beta(1+\delta_\beta), \gamma(1+\delta_\gamma)], \tag{3.10.330}$$

$$t_y = [\alpha'(1+\delta_{\alpha'}), \beta'(1+\delta_{\beta'}), \gamma'(1+\delta_{\gamma'})]), \tag{3.10.331}$$

where,

$$[\delta_\alpha, \delta_\beta, \delta_\gamma]^T = -H\Psi_x^{-1}(\vec{\omega}_x)|_{a,b,c=1}\vec{\nabla}(\vec{\omega}_x)|_{a,b,c=1}, \tag{3.10.332}$$

$$[\delta_{\alpha'}, \delta_{\beta'}, \delta_{\gamma'}]^T = -H\Psi_y^{-1}(\vec{\omega}_y)|_{a,b,c=1}\vec{\nabla}(\vec{\omega}_y)|_{a,b,c=1}. \tag{3.10.333}$$

We propose to apply the above iterations with the halting condition of $|\varepsilon_{t+1} - \varepsilon_t| \leq \varepsilon_\circ$, where $\varepsilon_\circ$ is the error resolution.

3.11 Content–based Pornography Filtering

3.11.1 Look–up Table–Based Approach

Assume that we are working in the color space $c_1c_2c_3$. Also, assume that c_1, c_2, and c_3 are linearly scaled and biased to give values in the interval $[0, 255]$ (8–bit representation). The pixel–based skin detection approach presumes that there exists a function $P : [0, 255]^3 \rightarrow [0, 1]$ for which, $P(\vec{c})$ shows the probability of $\vec{c}$ belonging to a skin–related area. The function P is also called the *skin probability map* (SPM) [69, 152]. Generally, P is cut by a fixed threshold to obtain a binary look–up table.

There are two general approaches for finding P, *ordinary look–up table* (OLUT) and *Baysian look–up table* (BLUT). Section 3.11.1 and 3.11.1 discuss the two methods for computing P. Finally, Section 3.11.1 proposed a method for detecting the skin map using a proper LUT.

Note that, some researchers try to code the LUTs using simple geometric rules [136, 333]. For example, rectangles [87], set of planes [147], and ellipses [270] are being used. We emphasize that those approaches are simplified versions of the general LUT–based skin detection process discussed here. As, for explaining the skin–tone locus by a set of geometric rules in the color domain, one first should prove that such locus exists.

Ordinary Look–up Table (OLUT)

The ordinary look–up table–based skin detection approach assumes that the SPM can be estimated from a proper training set. This approach is quite commonly used in the literature (*e.g.*, see [95, 317, 62, 402, 404, 403, 405, 325, 335, 199, 54, 83, 195, 408, 410, 406, 411, 409, 407, 316, 371, 141, 140, 136]).

Here, we investigate the validity of this assumption. Assume that we have two sets of 3–D vectors, relating to skin and non–skin (control), respectively. Also, assume that there are N_s pixels deliberately extracted from skin area and N_c pixels which do not represent skin. Both N_s and N_c should be large enough. Here, we have $N_s = 278530$ and $N_c = 192514$. Now, assume that H_s denotes the p–D histogram of all skin samples in an arbitrary color space ($p = 1, 2, 3$) (H_c is computed for the control dataset in a similar way). By setting different values of p we are able to remove one or two of the components of a color space. For computational purposes we select the number of bins in each direction equal to 16, 64, and 256 for 3–, 2–, and 1–D representations, respectively.

The main assumption behind the validity of OLUT–based skin–detection is that H_s can serve as P, the SPM. In this way, for an arbitrary threshold value θ, all color vectors $\vec{c}$ that satisfy $P(\vec{c}) \geq \theta$ are regarded as skin samples. Due to the fact that the summation of all elements of H_s equal unity, the elements of H_s may be too small. Thus, setting $P(\vec{c}) = H_s(\vec{c})$ will result in difficulties in selecting θ. To make the range

of feasible values of θ more appropriate, we compute P as,

$$P(\vec{c}) = \frac{H_s(\vec{c})}{\max_{\vec{d}}\{H_s(\vec{d})\}}. \tag{3.11.334}$$

In this way, θ can accept all values in the range of $[0, 1]$ [402].

For an arbitrary value of θ, the OLUT is a binary array with a size similar to H_s and H_c. Zero element in the $\vec{c}$ bin means that $\vec{c}$ does not represent skin tone, while unity indicates skin. Then, the *true positive* (TP) and *false positive* (FP) values are computed as,

$$TP = \sum_{\vec{c}\in[0,L]^p} H_s(\vec{c})LUT(\vec{c}), \tag{3.11.335}$$

$$FP = \sum_{\vec{c}\in[0,L]^p} H_c(\vec{c})LUT(\vec{c}). \tag{3.11.336}$$

We expect a value of θ resulting in a TP of around unity and a FP of around zero. Note that FP and TP are always between zero and unity.

For an arbitrary color space and a value of p, by selecting different values of θ in the $[0, 1]$ interval, we compute a set of corresponding (TP, FP) pairs. When these pairs are drawn in an axis, we reach to the *receiver operator characteristics* (ROC) curve. This curve is a keen way to determine an appropriate value of θ.

Baysian Look–up Table (BLUT)

Assume that a color vector $\vec{c}$ is given. Note that, what we actually need is $p(skin|\vec{c})$. From the Bayes theory we know that,

$$P(skin|\vec{c}) = \frac{P(\vec{c}|skin)p(skin)}{P(\vec{c}|skin)p(skin) + P(\vec{c}|\sim skin)p(\sim skin)}, \tag{3.11.337}$$

where $p(skin)$ and $p(\sim skin)$ are a priori probabilities, which are absolutely unknown. Here, $p(skin)$ and $p(\sim skin)$ are the probabilities that an arbitrary pixel represents skin and non–skin tones, respectively. Now, we write down the Bayes equality for $p(\sim skin|\vec{c})$ as,

$$P(\sim skin|\vec{c}) = \frac{P(\vec{c}|\sim skin)p(\sim skin)}{P(\vec{c}|skin)p(skin) + P(\vec{c}|\sim skin)p(\sim skin)}. \tag{3.11.338}$$

Note that,

$$P(skin|\vec{c}) + P(\sim skin|\vec{c}) = 1, \tag{3.11.339}$$

as trivially expected.

Using (3.11.337) and (3.11.338) we have,

$$\frac{P(skin|\vec{c})}{P(\sim skin|\vec{c})} = \frac{p(skin)}{p(\sim skin)} \times \frac{P(\vec{c}|skin)}{P(\vec{c}|\sim skin)}. \tag{3.11.340}$$

Here, we define two notations:

$$p^{-}(\vec{c}) = \frac{P(\vec{c}|skin)}{P(\vec{c}|\sim skin)}, \tag{3.11.341}$$

$$p^{+}(\vec{c}) = \frac{P(skin|\vec{c})}{P(\sim skin|\vec{c})}. \tag{3.11.342}$$

Note that according to (3.11.340), we have,

$$p^{+}(\vec{c}) = \frac{p(skin)}{p(\sim skin)} \times p^{-}(\vec{c}). \tag{3.11.343}$$

Using the normalization scheme used in (3.11.334), we have,

$$\frac{p^{+}(\vec{c})}{\max_{\vec{d}}\{p^{+}(\vec{d})\}} = \frac{p^{-}(\vec{c})}{\max_{\vec{d}}\{p^{-}(\vec{d})\}}, \tag{3.11.344}$$

which eliminates the two unknown probabilities of $p(skin)$ and $p(\sim skin)$. In fact, $p^{-}(\vec{c})$ is computed using two proper datasets of skin and non–skin, and (3.11.344) enables us to use it for creating a BLUT. From this point, everything is just similar to what performed in the OLUT–based skin detection process, described in Section 3.11.1. This approach is used in [329, 91, 67].

Skin Map Computation

Assume that we have the p–D color space X in which H is presumed to be a proper LUT for skin detection. Assume that H is a b^p–bin histogram. For the given image I, the process of finding its corresponding SPM is straightforward: compute J, the representation of I in X. Then, for each single pixel $\vec{c}$, compute the corresponding bin in H. Now, $\vec{c}$ represents skin iff the corresponding bin in H holds unity. This process results in the binary image M (M and I are in the same size). For the sake of simplicity, we propose another version of this scenario. For $0 < \lambda < 1$, first compute I_λ as the resized version of I with ratio λ ($\lambda = 10\%$ is a proper choice). Then, compute the SPM for I_λ as described above. Lets call the SPM of I_λ as M_λ. Now, up–sample M_λ, with ratio $\frac{1}{\lambda}$, to reach to M (which is the same size as I). According to the performance of MATLAB in processing arrays, and its deficiency in working with for loops, the above scheme gives considerably higher speed (about 4 times). The choice of the upsampling method depends on the expected quality of results; for fast evaluation the nearest neighborhood is a proper choice while bi–linear gives

LUTcensor

```
function M=LUTcensor(I,H,K,C,ds,method)
%Function LUT censor performs LUT-based censor on the image I.
%Usage: M=LUTcensor(I,H,K,C,ds,method);
%Inputs: I:Original Image.
%       H:The LUT.
%       K:Color space.
%       C:Channel (123,12,23,31).
%       ds:Downsampling ratio (default 0.1)
%       method: Upsampling method ([nearest],bilinear)
%Outputs: M:Censor map.
%
%Revision 1.1 Arash Abadpour, Spring 2005 (March 27 2005)
```

Figure 3.106: Implementation of the *LUTcensor* function.

smoother results in cost of higher computational complexity. Function *LUTcensor* finds the skin areas in an image according to a given LUT (see Figure 3.106).

3.11.2 Histogram–based Approach

For a set of color images all containing homogenous pixels, the color histograms in twelve of the color spaces stated in Section 2.1 are computed. The histograms of different images in a common color spaces are compared using the χ^2 distance, introduced in Section 3.11.2. Here, we investigate the original 3–D histograms along with three 2–D and three 1–D ones to analyze the effects of rejecting one or two components out of the color spaces.

chi2

```
function chi=chi2(h1,h2)
%Function chi square compute sthe chi square distance between two
%histograms h1 and h2.
%Usage: chi=Chi2(h1,h2);
%Inputs:     h1,h2:The given histograms.
%Outputs: chi:The chi square distance.
%Revision 1.0 Arash Abadpour Summer 2004 (1384/5/13)
```

Figure 3.107: Implementation of the *chi2* function.

Distance Function

Assume the two sets of d–dimensional vectors, I_1 and I_2 and the two corresponding m–bin histograms H_1 and H_2. The χ^2 distance between H_1 and H_2 is defined as:

$$\chi^2 = \frac{1}{2}\sum_{i_1=1}^{m}\cdots\sum_{i_d=1}^{m}\frac{\Big(H_1(i_1,\cdots,i_d)-H_2(i_1,\cdots,i_d)\Big)^2}{H_1(i_1,\cdots,i_d)+H_2(i_1,\cdots,i_d)} \tag{3.11.345}$$

Note that, for identical histograms, χ^2 gets zero, while non–crossing histograms are ranked with unity χ^2 distance. Hence, near zero values of χ^2 shows similarity.

The function *chi2* in **CIP** computes the χ^2 distance between two n–D distributions (see Figure 3.107). Also, the function *censor* finds the skin areas in an image using the χ^2 distance (see Figure 3.108).

censor

```
function [J,M]=censor(I,n,sig,US,thr)
%function censor censors an erotic image according to the signature in sig
%and the upsamplingmethod in US.
%Usage: [J,M]=censor(I,n,sig,US,thr);
%Input: I: Original image.
%       n: Tree Depth.
%       sig: Signature  (1: ssH, 2: ssa (default), 3: ssI_3)
%       US: Upsampling method (1: nearest (default), 2: bilinear, 3:
%       bicubic)
%       thr:Threshold.
%Output: J:Censored Image.
%        M:Skin Map.
%Revision 1.1 Arash Abadpour, winter 2005 (March 13, 2005)
```

Figure 3.108: Implementation of the *censor* function.

3.11.3 PCA–based Approach

The original idea of PCA–based skin detection comes from two essential facts from the color image processing literature. Firstly, there are experimental results that prove PCA to be a perfect tool for modelling homogenous regions in the color images of the nature. For example in [99], PCA–based approach is successfully applied for detecting homogenous regions in color images, or for extracting homogenous regions according to a given query. Secondly, there are physical evidences behind the performance of PCA in color image processing [268]. Hence, using proper skin and control datasets, an experiment is carried out. For a given color patch I, the PCA transform gives the

two vectors $\vec{\eta}$ and $\vec{v}$, the expectation and the direction of the first principal component of I, respectively. On the other hand, each pair of vectors $\vec{\eta}$ and $\vec{v}$ represent a unique line exists. We drew the lines corresponding to skin patches in bold, while the lines regarding non–skin patches, are drawn in dash. The result is shown in Figure 3.109. This shape reveals that all of the lines relating to skin patches pass through a small region of the color space, while non–skin patches manifest a more scattered pattern. We emphasize that this visual cue is backed with physical evidence [268]. If fact, the closest point to all skin–related lines gives the principal skin color. In this frame, any skin swatch contains a set of pixels having the same color as the principal skin color, transferred along an axis by lighting conditions, and scattered according to the noise.

Here, the proposed PCA–based skin detection method is introduced. Section 3.1.10 discussed the mathematics of finding the closes point to a set of lines. It is used in Section 3.11.3 for proposing a PCA–based skin detector. After discussing the way to select the threshold in Section 3.11.3, Section 3.11.3 introduces the proposed method for finding the skin map of a given image.

PCA–Based Skin Detection

Assume that the set of n swatches $I_1, \cdots, I_n$ are given as the skin training data. We propose to perform PCA on each $I_j, j = 1, \cdots, n$ to have $\vec{\eta}_i$ and $\vec{v}_i$. Here, $\vec{\eta}_i$ and $\vec{v}_i$ are two 3×1 vectors representing the expectation vector and the first principal direction of I_i. In this way, we have n lines $l_1, \cdots, l_n$. As such, l_i is parallel to $\vec{v}_i$ and

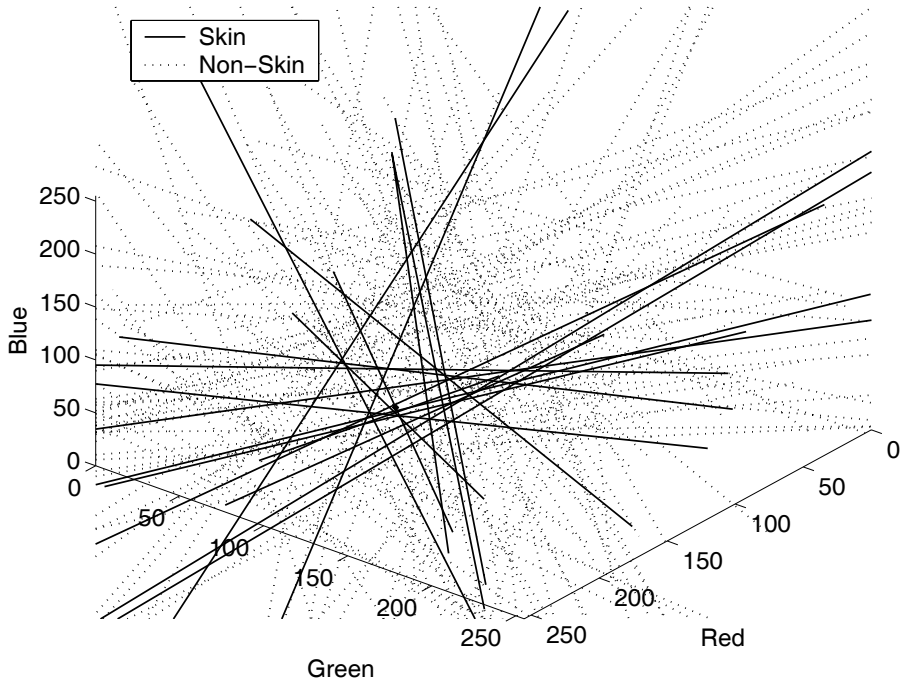


Figure 3.109: Lines corresponding to the skin and non–skin patches. Note the converging pattern of skin–related lines. Refer to the text for more details.

PCAdistance

```
function d=PCAdistance(I,x)
%Function PCADistance computes the histogram-based PCA distance between the
%data cluster in I and the point x. Everything occurs in R3.
%Usage:d=PCAdistance(I,x);
%Inputs: I:Given image.
%        x:3*1 vector.
%Outputs:d:Distance.
%Revision 1.01 Arash Abadpour Spring 2005 (19 April, 2005)
```

Figure 3.110: Implementation of the *PCAdistance* function.

passes through $\vec{\eta}_i$. Now, we propose to find the closes point to these n lines. Assume that $\vec{S}$ is the 3–D point computed as described in Section 3.1.10. Now, assume that the swatch I is given and the process should give s_I, the distance I to the class of all skin swatches. Performing PCA on I results in $\vec{\eta}_I$ and $\vec{v}_I$. Now, s_I is computed as the distance from $\vec{S}$ to the line $[\vec{\eta}_I, \vec{v}_I]$,

$$\delta_I^2 = \left\| (\vec{S} - \vec{\eta}_I) - \left[\vec{v}_I^T (\vec{S} - \vec{\eta}_I) \right] \vec{v}_I \right\|^2, \tag{3.11.346}$$

For a fixed value of θ, all swatches I are detected to be skin which satisfy $s_I < \theta$. Function *PCADistance* finds the distance from a specific point to a given line and function *PCACensor* finds the skin area in an image using a given principal color (see Figures 3.110 and 3.111).

PcaCensor

```
function [M,D]=PCACensor(I,n,X,thr,US)
%function PCA censor censors an erotic image according to the signature in
%x
%and the upsamplingmethod in US.
%Usage: [M,D]=PCACensor(I,n,x,thr,US);
%Input: I: Original image.
%       n: Tree Depth.
%       x: 3*a vector.
%       thr:Threshold.
%       US: Upsampling method (1: nearest (default), 2: bilinear, 3:
%       bicubic)
%Output: M:Skin Map.
%        D:Distance Map.
%Revision 1.1 Arash Abadpour, Spring 2005 (April 12, 2005)
```

Figure 3.111: Implementation of the *PcaCensor* function.

Determining the Threshold

As discussed in Section 3.11.3, having a proper value of θ, one can perform the proposed PCA–based skin detection process. One of the most important points of the proposed method is the appropriate selection of θ.

Assume that we have a dataset of n_s skin swatches $I_1, \cdots, I_{n_s}$ and a dataset of n_c non–skin (control) swatches $\tilde{I}_1, \cdots, \tilde{I}_{n_c}$. Also, assume that the vector $\vec{S}$ is computed from the skin dataset as described in Section 3.11.3. Now we compute the values of $s_1, \cdots, s_{n_s}$ and $\tilde{s}_1, \cdots, \tilde{s}_{n_s}$ as,

$$s_i = \delta_{I_i}, i = 1, \cdots, n_s, \tag{3.11.347}$$

$$\tilde{s}_i = \delta_{\tilde{I}_i}, i = 1, \cdots, n_c. \tag{3.11.348}$$

Now, for an arbitrary value of θ, $n_s(\theta)/n_s$ and $n_c(\theta)/n_c$ give the true positive (TP) and false positive (FP) rate of the skin detector with the threshold selected as θ. Here, $n_s(\theta)$ is the number of s_i satisfying $s_i < \theta$. $n_c(\theta)$ is defined in a similar way for $\tilde{s}_i$. Using different values of θ in the range of $[0, 1]$ and drawing the (TP, FP) pairs gives the receiver operator character (ROC) curve. From this curve one can find a proper value of θ resulting in reasonable values of TP and FP.

Skin Map Computation

For an image I, there exists a corresponding skin probability map (SPM) S_I. Here, for each pixel $\vec{I}_{xy}$, S_{Ixy} is zero or one, determining wether $\vec{I}_{xy}$ represents skin or not.

Although, this definition relates to pixel–based skin detection literature, a similar representation is adoptable here.

Assume that the $H \times W$ image I is given. Also, assume that the integer value of $\lambda > 1$ is given. Divide I into non–overlapping $\lambda \times \lambda$ swatches and perform the skin detection task on each swatch, as discussed in section 3.11.3. The result is a $\frac{H}{\lambda} \times \frac{W}{\lambda}$ binary SPM ($S_{\lambda I}$). Now, up–sample $S_{\lambda I}$, with ratio $\frac{1}{\lambda}$, to reach to S_I (which is the same size as I).

3.12 Video Segmentation and Change Detection

Let two images I_1 and I_2 to belong to the same scene in slightly different times. Then each pixel in I_1 and I_2 is an m–D realization. Also, let image I_1 to be segmented into c classes of $\phi_i, i = 1, \cdots, c$ using the proposed FPCAC method (see Section 3.5.2). Here, $J_{ixy}, i = 1, \cdots, c$ shows the membership of $\vec{I}_{1xy}$ to the i–th class.

Now, perform the FPCA on the fuzzy set,

$$\tilde{X} = \left\{ \left(\vec{I}_{2xy}; J_{ixy}^m \right) \middle| 1 \leq x \leq W, 1 \leq y \leq H \right\}, \tag{3.12.349}$$

to find the new clusters $\tilde{\phi}_i, i = 1, \cdots, c$. In fact, we are using the temporal redundancy of successive frames, assuming that the fuzzy membership of a pixel to the c classes remains still if there is no abrupt change. The reason behind finding the new clusters in I_2 is to compensate probable slight changes corresponding to the lighting and the motion. Now, we have the new membership values $\tilde{J}_{ixy}$, which show the level of

membership of $\vec{I}_{2xy}$ to the i–th new class $\tilde{\phi}_i$.

We propose computing,

$$\delta_{xy} = \frac{1}{c}\sqrt{\sum_{i=1}^{c} J_{ixy}(J_{ixy} - \tilde{J}_{ixy})^2}, 1 \leq x \leq W, 1 \leq y \leq H, \tag{3.12.350}$$

as the probability of the point (x, y) being changed from I_1 to I_2. In fact, δ_{xy} measures the net amount of change in membership of pixels to the classes in the successive frames. Note that while these fuzzy change values are computed, the clusters are also updated at the same time.

If $I_1 \equiv I_2$, then J_{ixy} and $\tilde{J}_{ixy}$ will be identical, resulting in δ_{xy} being zero everywhere, as desired. Now, assume that there is no change between the two images I_1 and I_2, unless for the changes in the imaging conditions. Assume that $\vec{x}_i$ and $\vec{y}_i$ are the color vectors of the same physical points in the two images I_1 and I_2, respectively. We model the change in imaging conditions as a linear operation [268]. Assume that $\vec{x}_i$ and $\vec{y}_i$ relate through a linear transform namely,

$$\vec{x}_i = A\vec{y}_i + \vec{b}. \tag{3.12.351}$$

Here, we model A as a non–singular invertible matrix with its eigenvalues being almost constant. This situation relates tthe o cases that the color axes rotate (change in chromaticity of the illumination), scale (change in achromaticity of the illumination), and translate. The model restricts unbalanced scaling of color components which changes the color information non–meaningfully (for details see [268]). The matrix A

in the *singular value decomposition* (SVD) form is written as,

$$A = VDU^{-1}. \tag{3.12.352}$$

where, U and V are orthogonal matrices and D is a diagonal matrix with the eigenvalues of A as its elements.

The expectation vectors in the two images I_1 and I_2 relate as,

$$E\Big\{\vec{x}_i\Big\} = E\Big\{A\vec{y}_i + \vec{b}\Big\} = AE\Big\{\vec{y}_i\Big\} + \vec{b}. \tag{3.12.353}$$

The fuzzy covariance matrices of the two images I_1 and I_2 satisfy the following equations,

$$C_1 = AE\Bigg\{\Big(\vec{y}_i - E\{\vec{y}_j\}\Big)\Big(\vec{y}_i - E\{\vec{y}_j\}\Big)^T\Bigg\}A^T = AC_2A^T. \tag{3.12.354}$$

Assume that the eigenvectors of C_1 are $\vec{v}_i, i = 1, \cdots, m$ corresponding to the eigenvalues of $\lambda_i, i = 1, \cdots, m$ and the eigenvectors of C_2 are $\vec{u}_i, i = 1, \cdots, m$ corresponding to the eigenvalues of $\rho_i, i = 1, \cdots, m$. Also, assume the eigenvectors of A to be $\vec{w}_i, i = 1, \cdots, m$ corresponding to the eigenvalues of $\varepsilon_i, i = 1, \cdots, m$. Thus,

$$\forall i, C_1\vec{v}_i = \lambda_i\vec{v}_i, C_2\vec{u}_i = \rho_i\vec{u}_i, A\vec{w}_i = \varepsilon_i\vec{w}_i. \tag{3.12.355}$$

First assume that the eigenvectors of A are all exactly equal to the fixed value of λ (or equivalently $\forall i, \varepsilon_i = \lambda$). Thus,

$$A = VDU^{-1} = Vdiag(\lambda, \cdots, \lambda)U^{-1} = \lambda VU^{-1}. \tag{3.12.356}$$

In this situation,

$$A^T = \lambda U V^{-1} = \lambda^2 A^{-1} \rightarrow A^T A = A A^T = \lambda^2 I. \tag{3.12.357}$$

Now, note that,

$$C_1 A\vec{u}_i = A C_2 A^T A\vec{u}_i = \lambda^2 A C_2 \vec{u}_i = \lambda^2 \rho_i A\vec{u}_i. \tag{3.12.358}$$

Thus, $A\vec{u}_i$ is the eigenvector of C_2 corresponding to the eigenvalue of $\lambda^2\rho_i$. Note that,

$$\|A\vec{u}_i\| = \lambda \|\vec{u}_i\| = \lambda. \tag{3.12.359}$$

As the eigenvalues and eigenvectors of a single matrix are identical, we have,

$$\Big\{(\frac{1}{\lambda} A\vec{u}_1, \lambda^2\rho_1), \cdots, (\frac{1}{\lambda} A\vec{u}_m, \lambda^2\rho_m)\Big\} = \Big\{(\vec{v}_1, \lambda_1), \cdots, (\vec{v}_m, \lambda_m)\Big\}. \tag{3.12.360}$$

As $\lambda^2 > 0$ we have,

$$\forall i, \vec{v}_i = \frac{1}{\lambda} A\vec{u}_i, \lambda_i = \lambda^2 \rho_i. \tag{3.12.361}$$

Thus, using the above re–clustering method, the cluster $\phi = [\vec{\eta}, \vec{v}]$ in I_2 results in the cluster $\tilde{\phi} = [A\vec{\eta} + \vec{b}, A\vec{v}]$. Now, we have,

$$\Psi(\vec{x}_i, \tilde{\phi}) = \left\| \Big[(A\vec{y}_i + \vec{b}) - (A\vec{\eta} + \vec{b})\Big] - \frac{1}{\lambda^2} \vec{v}^T A^T \Big[(A\vec{y}_i + \vec{b}) - (A\vec{\eta} + \vec{b})\Big] A\vec{v} \right\|^2 \tag{3.12.362}$$

$$\Psi(\vec{x}_i, \tilde{\phi}) = \left\| A(\vec{y}_i - \vec{\eta}) - \vec{v}^T (\vec{y}_i - \vec{\eta}) A\vec{v} \right\|^2 = \lambda^2 \Psi(\vec{x}_i, \tilde{\phi}), \tag{3.12.363}$$

$$\tilde{J}_{ixy} = \frac{\Psi(\vec{x}_i, \tilde{\phi}_j)^{-\frac{1}{m-1}}}{\sum_{k=1}^{c} \Psi(\vec{x}_i, \tilde{\phi}_k)^{-\frac{1}{m-1}}} = \frac{\Psi(\vec{x}_i, \phi_j)^{-\frac{1}{m-1}}}{\sum_{k=1}^{c} \Psi(\vec{x}_i, \phi_k)^{-\frac{1}{m-1}}} = J_{ixy}, \tag{3.12.364}$$

resulting in $\delta_{xy} = 0$. Thus, the proposed method will be independent of the lighting and imaging conditions.

Now, assume a more realistic case that ε_is are not exactly the same but we have $\lambda - \delta\lambda \leq \varepsilon_i \leq \lambda + \delta\lambda$. For the cases that $\frac{\delta\lambda}{\lambda}$ is too small the above equations change to semi–equations and still marginally hold. In this situation $\delta_{xy} \simeq 0$.

In contrast, physical changes result in different material in a single point in different shots. Hence, they produce absolutely different values of J_{ixy} and $\tilde{J}_{ixy}$, resulting in non–zero patterns of δ_{xy}. In the proposed method, at the same time both the image sequence segmentation and the fuzzy change detection are performed.

Here, we integrate the concepts developed in last sections to propose am efficient video segmentation and fuzzy change detection method. In fact, the segmentation method developed in Section 3.5.2 roles as an initializer in the first frame and the change detection method proposed in Section 3.12, both generalizes the segmentation results to the next frames and gives the fuzzy change detection mask.

Assume that, the image sequence $I_1, \cdots, I_m$ is given. Also, assume that the parameter c denotes the approximate number of color clusters in a typical frame of the sequence. The two other parameters of the proposed clustering method are δ (the convergence threshold) and k (the convolution kernel radius). These two parameters do not have serious effect on the final results. In all cases we set $\delta = 0.05$ and $5 \leq k \leq 10$. Smaller values of δ will negligibly increase the elapsed time and varying

k affects the boundary of the segments. Setting $k = 1$ (no smoothing) results in segments with many holes and large values of k give over–smoothed segments. In various tests, both on still images and image sequences, it is observed that the above values of δ and k are appropriate.

Assume that, applying the proposed segmentation method introduced in Section 3.5.2 on I_1 has resulted in the fuzzy membership maps of $J_{1i}, i = 1 \cdots c$. We propose applying the change detection method proposed in Section 3.12 on I_2 to gain new maps of $J_{2i}, i = 1 \cdots c$ and the change map δ_2. Iterating the same operation on all consecutive frames of the image sequence, results in the change maps of $\delta_i, i = 2 \cdots n$ and fuzzy membership maps of $J_{ji}, j = 1 \cdots m, i = 1 \cdots c$. Here, δ_i serves as the desired fuzzy change detection mask and J_{ij} holds the fuzzy segmentation result corresponding to the i–th frame.

To enhance the results, we avoid working on each frame of the image sequence. In this way we reject frames in a regular tender. Instead, we work on one out of each τ consecutive frames. Although τ can be left unity, according to the displacement speed of the objects in the image sequence, different values of τ can be used to boost the time performance. We measure two values of t_0 and t_f as the time needed to process the first frame (initialization time) and the time needed to apply the change detection method on each frame, respectively. The proposed method is utilized in *singlecd* shown in Figure 3.112.

singlecd

```
function [k,t,K1,K2]=singlecd(I1,I2,C,m,kernel)
%Single change detection gives the zones of the change inthe two images
%using PCA.
%
%Usage: [k,t,K1,K2]=singlecd(I1,I2,C,m,kernel);
%Inputs: I1,I2:Input images.
%        C:Cluster number.
%        m:Fuzziness.
%        kernel:Convlusion kernel size.
%Outputs: k:change map.
%         t:elapsed time.
%         K1,K2:Related segmentation data.
%
%Revision 1.0 (c) Arash Abadpour Fall 2004 (Nov. 14, 2004)
```

Figure 3.112: Implementation of the *singlecd* function.

3.13 Image Sequence Recoloring

3.13.1 Proposed Method

In the PCA–based recoloring method discussed in Section 3.9.2, a fuzzy blending approach is utilized to make the recolored images more realistic. The lack of such mixing process will result in non–satisfactory edges. In that methodology, a few color categories are given by the user (as an actual set of corresponding swatches in the two images). Then, each pixel is recolored given that it completely belongs to each color category. After that stage, the blending process combines these alternatives for each individual pixel to form the final recolored vector. In Section 3.9.2 we used a normalization factor along with a parametric fuzzy membership to produce the blending weights. Here, we use the blending idea equipped with a novel fuzzification method. The new fuzzification method depends on only one meaningful parameter in contrast with the two–parameter Butterworth function used in Section 3.9.2.

Assume that the reference image I_r and the source image sequence $I_i, i = 1 \cdots, n$ are given. Also, assume that the set of c corresponding swatches of $[\tilde{r}_i, r_i], i = 1, \cdots, c$ highlight homogenous patches that should rule the recoloring process. As such, for $1 \leq i \leq c$, the $\tilde{r}_i$ and r_i are the corresponding regions in I_r and I_1, respectively (see Figure 3.113).

Figure 3.113 shows a sample process of performing the proposed recoloring method on a single–frame image sequence. Figure 3.113–a and Figure 3.113–b show the source

and the reference images, respectively. Both images contain two homogenous swatches which are given by the user to guide the recoloring process. Figure 3.113–c and Figure 3.113–d illustrate the fuzzy membership maps of the source image according to the two swatches. Figure 3.113–e and Figure 3.113–f show the single–swatch recoloring results and Figure 3.113–h shows the final result of the proposed recoloring process. The total time elapsed on producing this result is $1.8s$. Here, the value of m (fuzziness) is 2, but no difference is observed for other values of m. Note the high quality of the results of the proposed method along with its computational performance. Also, note the naturalistic contact of the sky and grasses at the horizon.

Here, we compute $\vec{\tilde{\eta}}_i$, $\tilde{V}_i$, and $\vec{\tilde{v}}_i$ as the expectation vector, PCA matrix, and the first principal direction of $\tilde{r}_i$, respectively. In the same way, $\vec{\eta}_i$, V_i, and $\vec{v}_i$ are computed for r_i. Let $\vec{c}$ denote an arbitrary color vector in the i–th frame I_i. In Section 3.9.2 we used the color vector $\vec{c}_j$ as the recolored version of $\vec{c}$, according to the reference swatches $[\tilde{r}_j, r_j]$, as,

$$\vec{c}_j = V_j \tilde{V}_j^T (\vec{c} - \vec{\tilde{\eta}}_j) + \vec{\eta}_j. \tag{3.13.365}$$

Computation of the color vector $\vec{c}_j$ in (3.13.365) is based on the assumption that $\vec{c}$ completely belongs to the j–th color category. This transformation is called single–swatch–recoloring (see Section 3.9.2).

The idea of blending recolored vectors corresponding to a single pixel is the novel contribution of *Reinhard et al.* [297]. In that methodology, each color vector $\vec{c}$ belongs

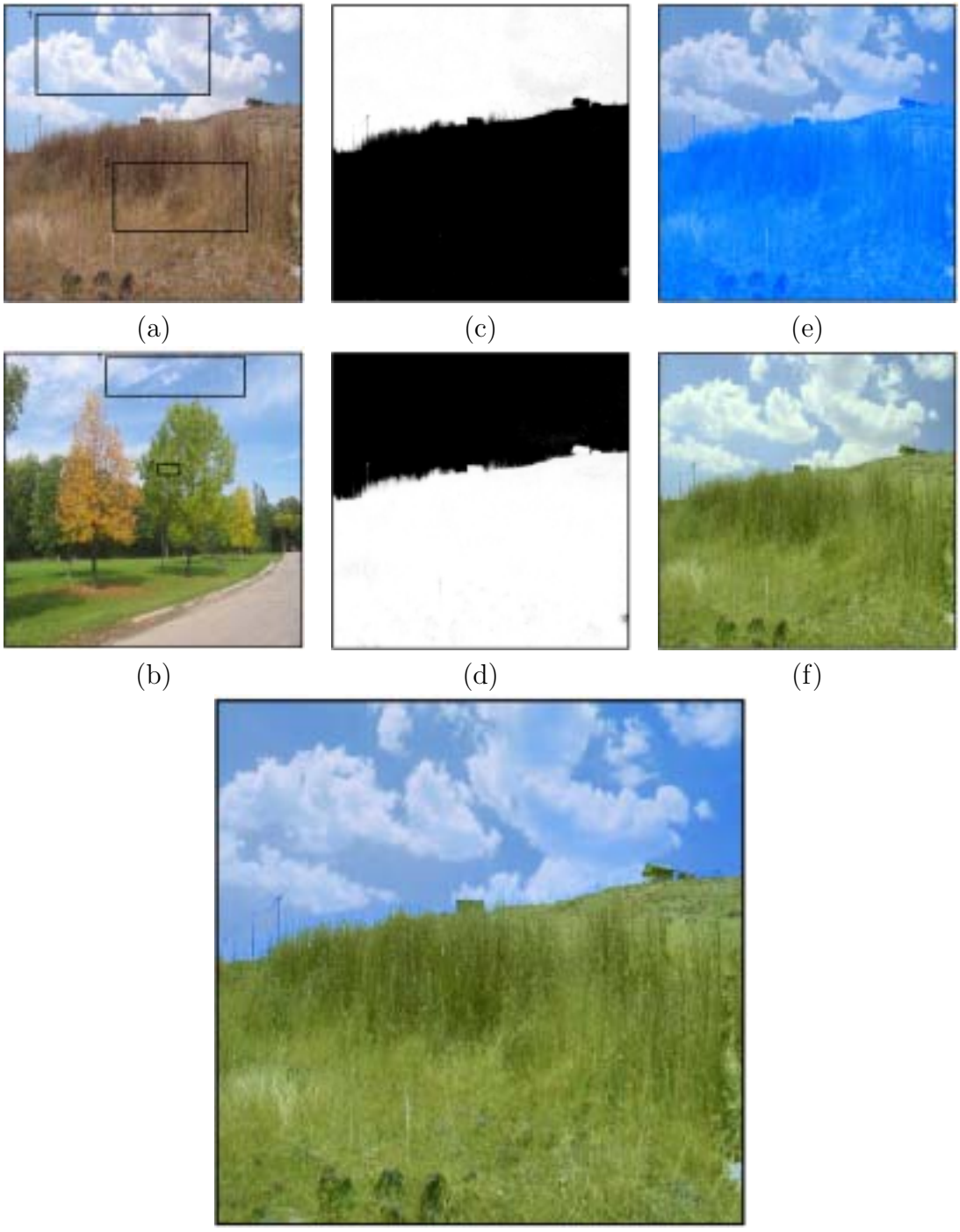


Figure 3.113: Result of the proposed recoloring process for a single–frame sequence. (a) Source image. (b) Reference image. (c) and (d) Fuzzy membership maps. (e) and (f) Results of single–swatch recoloring. (g) Final result elapsing less than 2 seconds.

to each color category (or color swatch) to some extent. Let's denote the membership of $\vec{c}$ to $\tilde{r}_i$ by $p_{\vec{c},i}$, for $i = 1, \cdots, c$. The membership values should satisfy $\sum_{i=1}^{c} p_{\vec{c},i} = 1$ for each pixel. Now, the recolored version of $\vec{c}$ is given by,

$$\vec{\tilde{c}} = \sum_{i=1}^{c} p_{\vec{c},i} \big[V_i \tilde{V}_i^T (\vec{c} - \vec{\tilde{\eta}}_i) + \vec{\eta}_i \big]. \tag{3.13.366}$$

The membership values of $p_{\vec{c},i}$ depend on the relative distances from $\vec{c}$ to the fixed color clusters.

Here, we propose to use the distance–based fuzzification approach used in fuzzy clustering methods like Fuzzy C–means (FCM) [168] and fuzzy principal component analysis–based clustering (FPCA) (see Section 3.5.2) to obtain the membership coefficients $p_{\vec{c},i}$. Assume that for an arbitrary color vector $\vec{c}$, the $d_i, i = 1, \cdots, c$ gives the LPRE distance from $\vec{c}$ to $\tilde{r}_i$, or equivalently $p_{\vec{c},i} = \tau_{r_i}(\vec{c})$. We propose to calculate the coefficients as,

$$p_{\vec{c},i} = \frac{d_i^{-\frac{1}{m-1}}}{\sum_{j=1}^{c} d_j^{-\frac{1}{m-1}}}, \quad m > 1, \tag{3.13.367}$$

where, m is the fuzziness. Assume that for $i \neq i'$, $d_i \leq d_{i'}$. Thus,

$$d_i^{\frac{1}{m-1}} \leq d_{i'}^{\frac{1}{m-1}} \rightarrow d_i^{-\frac{1}{m-1}} \geq d_{i'}^{-\frac{1}{m-1}}, \tag{3.13.368}$$

resulting in,

$$d_i \leq d_{i'} \leftrightarrow p_{\vec{c},i} \geq p_{\vec{c},i'}. \tag{3.13.369}$$

Hence, the membership values inversely relate to the corresponding distance functions (as desired). Also, note that the unity constraint is satisfied by $p_{\vec{c},i}, i = 1, \cdots, c$

FPCAFuzzificate

```
function K=FPCAFuzzificate(I,E,V,m,K1,K2)
%Function FPCAFuzzificate fuzzificates the given image due to the clusetr
%settings.
%Usage:      K=FPCAFuzzificate(I,E,V,m,K1,K2)
%Inputs:     I:Original image.
%           E,V:Cluster information.
%           m:Fuzziness.
%           K1: After emphasize filter size (0: ignore).
%           K2: Before emphasize filter size (0: ignore).
%Outputs:    K:Fuzzy maps.
%Revision 1.0 Arash Abadpour Fall 2004 (Nov. 27, 2004)
```

Figure 3.114: Implementation of the *FPCAFuzzificate* function.

defined in (3.13.367). This fuzzification scheme is developed in the function *FPCA-Fuzzificate* (see Figure 3.114).

Furthermore, we withdraw the constant denominator and write,

$$p_{\vec{c},i} \propto d_i^{-\frac{1}{m-1}}, \tag{3.13.370}$$

where, $x \propto y$ means there is a constant c satisfying $x = cy$. Now, the denominator appears as a constant dividing factor. As m approaches from 1^+ to ∞ the term $-\frac{1}{m-1}$ goes from $-\infty$ to 0^-. Thus, as m approaches 1^+, the fuzzification scheme proposed in (3.13.367) changes to a max–norm (ranking the smallest distance with unity membership and rejecting all others). Figure 3.115 illustrates the fuzzification results for $c = 2$ (two clusters). In this figure, the fuzzy membership of an artificial

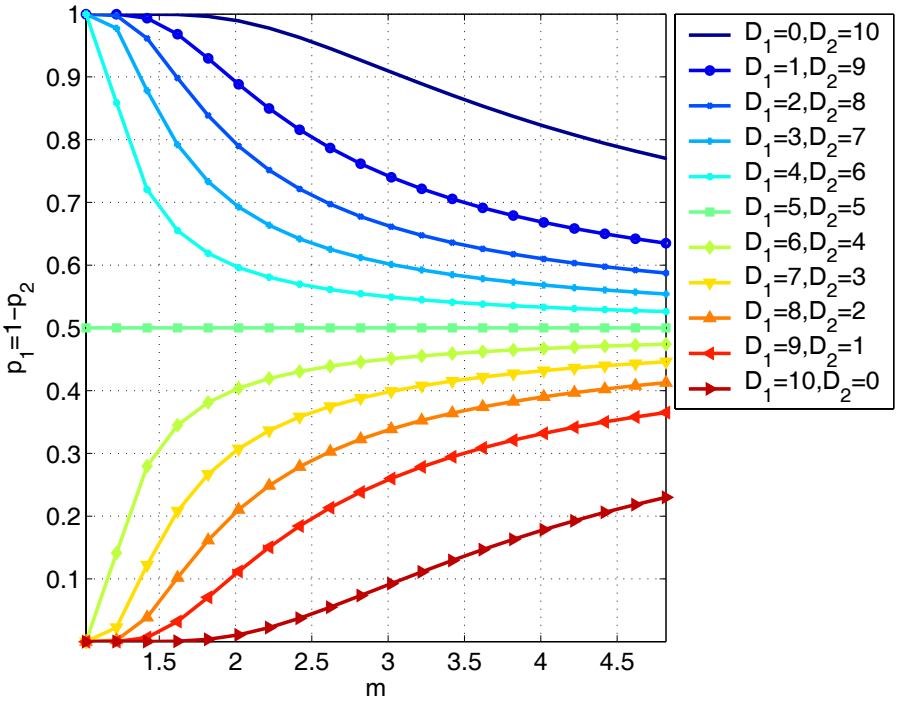


Figure 3.115: Fuzzification results of the proposed method for two–cluster case with different distances pairs and different values of m.

sample to the first cluster ($p_1 = 1 - p_2$) for 11 sets of distance values and for values of $1 < m \leq 5$ are shown. Note that the case of $D_1 = D_2$ results in $p_1 = p_2 = \frac{1}{2}$ independent of the value of m. Also, when m equals to 2, the fuzzy membership values are proportional to the reciprocal of distances. On the other hand, when m gets large, fuzzification tends to fade all membership values to $\frac{1}{2}$. Thus, the value of m controls the fuzzification sharpness. As a conclusion, $m = 1^+$ gives crisp results while $m \to \infty$ produces complete equilibrium (which is not meaningful).

Thus, the proposed image sequence–recoloring first asks user for two sets of corresponding swatches in the reference image and the first frame. Then, it performs the same task for each frame(see Figure 3.116). For the j–th frame the process contains:

- Computing the distance from each pixel to each of the c given swatches in the first frame.
- Computing fuzzy membership values using the distance values.
- Computing the recolored version of each pixel according to each of the c pairs of corresponding swatches.
- Blending the above c vectors according to the computed fuzzy membership values.

3.13.2 Computational Cost of the Available Methods for Still Image Recoloring

In this section, we determine the computational cost of the available recoloring methods. In all cases, the source and reference images are $W \times H$ color images and the swatches (in swatch–based methods) are a set of c pairs of $w \times h$ patches. Refer to the appendix C for a comprehensive discussion of the computational cost of operations discussed here. Table 3.5 lists the approximate cost of processing one pixel in each of the four available method.

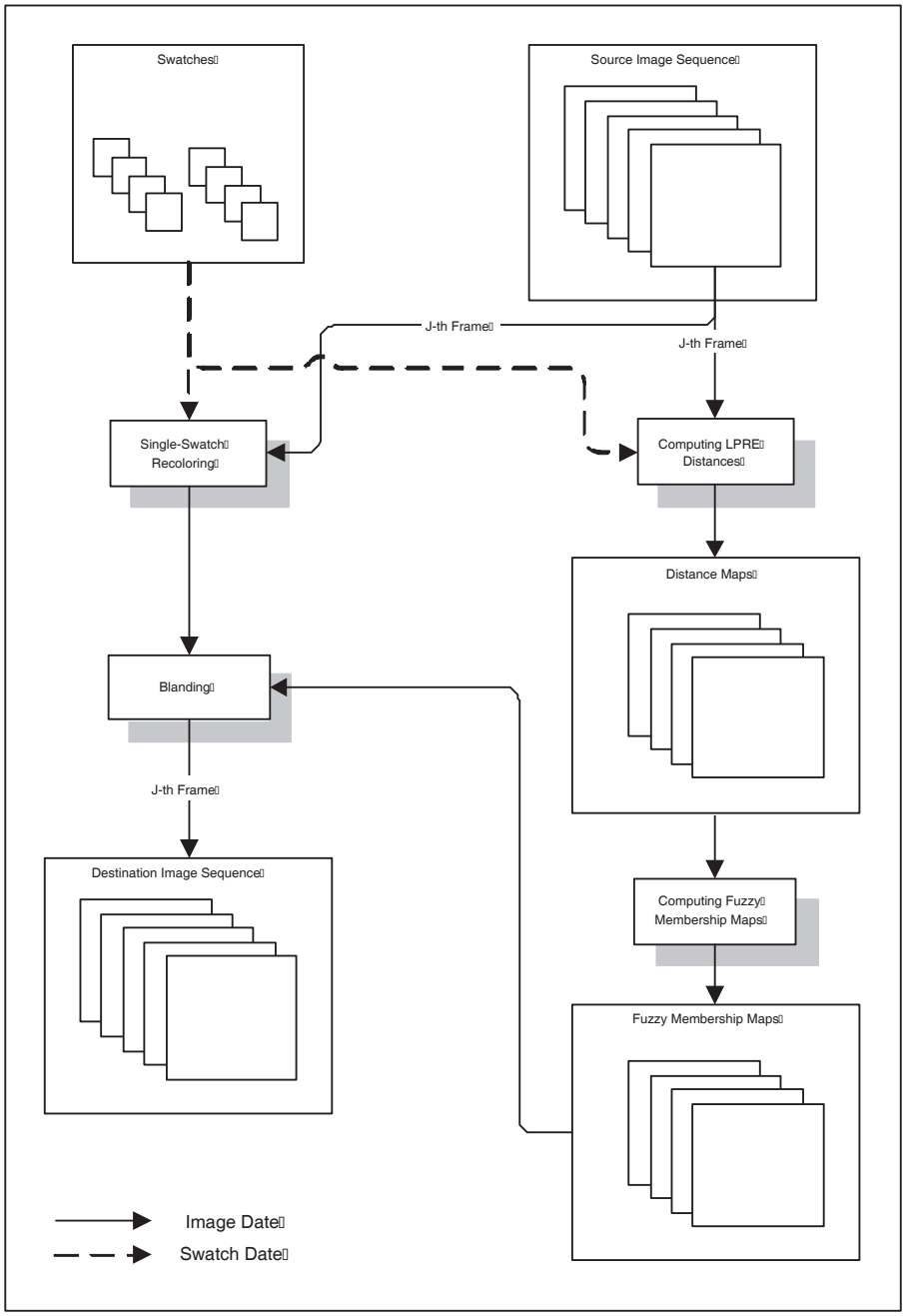


Figure 3.116: Flowchart of the proposed image sequence recoloring.

Method in [297]

The method in [297] works in $\ell\alpha\beta$ [305]. Converting one vector from RGB to $\ell\alpha\beta$ needs two vector by matrix multiplications and three logarithms. The inverse conversion needs two vector by matrix multiplications and three exponentials. In this way,

$$t_{RGB\to\ell\alpha\beta} = t_{\ell\alpha\beta\to RGB} = 2 \times 9 + 3 = 21, \tag{3.13.371}$$

flops. The process has to convert $WH + cwh$ vectors from RGB to $\ell\alpha\beta$. The total number of flops elapsed in this stage is,

$$t_1 = (WH + cwh)t_{RGB\to\ell\alpha\beta} = 21WH + 21cwh, \tag{3.13.372}$$

flops. After that, it needs to compute the standard deviation and expectation vector of the three components of $2c$ sets of wh 3×1 vectors. This process takes,

$$t_2 = 3 \times 2c(3wh + 2) = 18cwh + 12c, \tag{3.13.373}$$

flops. Then a process similar to single–swatch recoloring begins, which for each component of each pixel's color vector, performs the following process, c times,

$$y = \frac{\sigma_2}{\sigma_1}(x - \eta_1) + \eta_2 = \frac{\sigma_2}{\sigma_1}x + \left(\eta_2 - \frac{\sigma_2}{\sigma_1}\eta_1\right). \tag{3.13.374}$$

This process needs one flop for each component, or equivalently three flops for each pixel and each pair of swatches. Adding up all pixels and all pairs the total flops

elapsed in these stage equals,

$$t_3 = 3cWH + 2c, \tag{3.13.375}$$

flops. In this stage, for each pixel, the process have to compute fuzzy membership of each pixel to each category ($6c$ flops for the Euclidean distances and c flops for computing the reciprocals) and then blend the vectors ($4c+3$ flops) to form the final vectors. This makes a total of,

$$t_4 = WH(6c + c + 4c + 3) = 11cWH + 3WH, \tag{3.13.376}$$

flops for the whole image. Finally, the process returns WH vectors from $\ell\alpha\beta$ to $\ell\alpha\beta$,

$$t_5 = WHt_{\ell\alpha\beta\rightarrow RGB} = 21WH, \tag{3.13.377}$$

flops. Adding up all stages, the total number of flops needs to perform the recoloring by the method in [297] equals,

$$\begin{aligned} t = 21WH + 21cwh + 18cwh + 12c + 3cWH + 2c + 11cWH + \\ 3WH + 21WH = (45 + 14c)WH + 39cwh + 14c, \end{aligned} \tag{3.13.378}$$

flops which linearly depends on the image size.

Method in [92]

Unfortunately, the method in [92] is not completely described. Due to the fact that it contains hierarchical segmentation of both the source image and the reference image,

and according to the tree–like segmentation method utilized in that approach, it seems that the elapsed time of the process is at least dependent on the square of pixel count. We have no access to extra information regarding the complexity of the process and will leave it here.

Method in [156]

The algorithm in [156] works in $CIE-Lab$ color space. Converting a single pixel from RGB to $CIE-Lab$ (or vice versa) takes 26 flops. Thus, the algorithm first have to spend,

$$t_1 = 2 \times 26 \times WH = 52WH, \tag{3.13.379}$$

flops for converting both source and reference images to $CIE-Lab$ representation. Then it has to produce the 11 loci in each of the source and reference images. In this method, each locus is realized using a set of planes. Although, the actual number of planes is not restricted in [156], it seems from the figures of that paper that each locus shapes up by the intersection of at least one hundred planes. Also, we assume that producing each locus elapses number of flops equal to ten times the number of its members (a very optimistic figure). As such, producing the loci in the two images gets at least,

$$t_2 = 2 \times 11 \times 10WH = 220WH, \tag{3.13.380}$$

flops. At this point, for each given pixel, the algorithm should find a complex geometrical measure containing computation of intersection of a line with each of the one hundred planes surrounding each locus. Assuming that finding the intersection of a line and a plane gets 10 flops, this stage takes at least,

$$t_3 = 2WH \times 100 \times 10 = 2000WH, \tag{3.13.381}$$

flops. The subsequent geometrical transformation gets 5 flops for each pixel,

$$t_4 = 5WH, \tag{3.13.382}$$

flops. Finally, the process has to spend another $26WH$ flops for converting the resulting image back into RGB representation. Adding up all flops gives a sum of,

$$t = 52WH + 220WH + 2000WH + 5WH + 26WH = 2303WH, \tag{3.13.383}$$

flops, which is of course linearly related to the image size but much larger (about twenty times larger) than that of the method in [297].

Method in Section 3.9.2

Assume performing the method discussed in Section 3.9.2 with c pairs of corresponding swatches. The process first needs c times repeating the single–swatch recoloring which takes,

$$t_1 = c(12WH + 12wh + 159), \tag{3.13.384}$$

flops. After that the process has to compute the LPRE distance from each pixel to each of the c classes (the parameters of the classes are already available). It takes,

$$t_2 = 15cWH, \tag{3.13.385}$$

flops to do so. Then, the process have to compute c homogeneity factors corresponding to the swatches in the source image and divide each LPRE distance by the regarding homogeneity factor. This takes,

$$t_3 = (22wh + 1)c + cWH, \tag{3.13.386}$$

flops.

After that, the Butterworth function should be applied on each of the normalized distances. It takes,

$$t_4 = 4cWH, \tag{3.13.387}$$

flops. Finally, the process has to perform WH times fuzzy expectation for sets with c members taking,

$$t_5 = WH(4c + 3), \tag{3.13.388}$$

flops. Adding up the figures gives a whole of,

$$\begin{aligned} t = t_1 + t_2 + t_3 + t_4 + t_5 = c(12WH + 12wh + 159) + 15cWH + \\ (22wh + 1)c + cWH + 4cWH + WH(4c + 3) = \\ = (36c + 3)WH + 34cwh + 160c, \end{aligned} \tag{3.13.389}$$

flops. Note that t linearly depends on the image size.

Proposed Method

Like the method discussed in Section 3.9.2, the proposed method first needs to perform c times single–swatch recoloring, which takes,

$$t_1 = c(12WH + 12wh + 159), \tag{3.13.390}$$

flops. After that the process has to compute the LPRE distance from each pixel to each of the c classes (the parameters of the classes are already available) and compute their $-\frac{1}{m-1}$ power. It takes,

$$t_2 = 16cWH, \tag{3.13.391}$$

flops. The final step is performing WH times fuzzy expectation for sets with c members taking,

$$t_3 = WH(4c + 3), \tag{3.13.392}$$

flops. As such, the total number of flops equals,

$$\begin{aligned} t = t_1 + t_2 + t_3 &= c(12WH + 12wh + 159) + 16cWH + WH(4c + 3) \\ &= (32c + 3)WH + 12whc + 159c, \end{aligned} \tag{3.13.393}$$

flops, which is linearly related to the image size.

Table 3.5: Approximate cost of processing one pixel in different recoloring methods.

Method	**Approximate cost of processing one pixel**
[297]	$45 + 14c$
[92]	Not Available.
[156]	2303
Section 3.9.2	$36c + 3$
Proposed Method	$32c + 3$

Chapter 4

Experimental Results and Discussions

All algorithms are developed in *MATLAB 6.5* using *image processing toolbox 3.2* (*Windows xp* environment). The platform on which the codes are executed were two *1100 MHZ Pentium III* and *2600 MHz Pentium IV* personal computers, both with 256MB of RAM. The code of different algorithms are gathered in a package, called *color image processing* (CIP) toolbox (see Appendix A).

In practice, *WinZip 7.0 SR-1 (1285)* by *Nico Mark Computing, Inc.* is used as the compressor of binary and indexed arrays, when needed. Also, in many situations *Adobe Photoshop 6.0* is used as an image processing utility. *Advanced Batch Converter* 3.9 is used in this thesis as the JPEG2000 compression module and many conventional image processing processes are utilized using *ACDSee* 5.0.

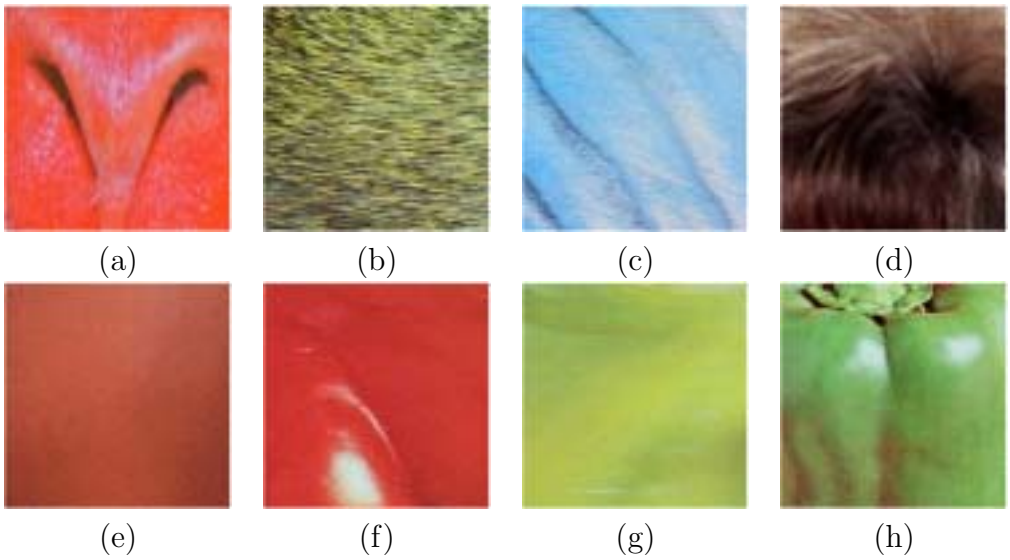


Figure 4.1: Eight homogenous regions in some standard images used for evaluating the proposed dimension reduction method. (a) Nose of *Mandrill*, (b) Wool of *Mandrill*, (c) Lateral part in nose of *Mandrill*, (d) Hair of *Girl*, (e) Skin of *Lena*, (f) Red pepper of *Peppers*, (g) Yellowish–Green pepper of *Peppers*, (h) Green pepper of *Peppers* (images are not proportionally resized).

4.1 Dimension Reduction in Color Fields

To evaluate the energy compaction of the pc_i channels, a large set of homogenous swatches were extracted from the test images. Figure 4.1 shows eight samples adopted from parts of the four standard images of *Lena*, *Girl*, *Peppers*, and *Mandrill*. The test swatches are converted to the proposed 1–D representation and a few typical results are shown in Figure 4.2. Numerical results are listed in Table 4.1.

As shown in Table 4.1, in contrast with the original color components with κ values around $\frac{1}{3}$, in PCA channels, the information mostly gathers in the first channel (0.88

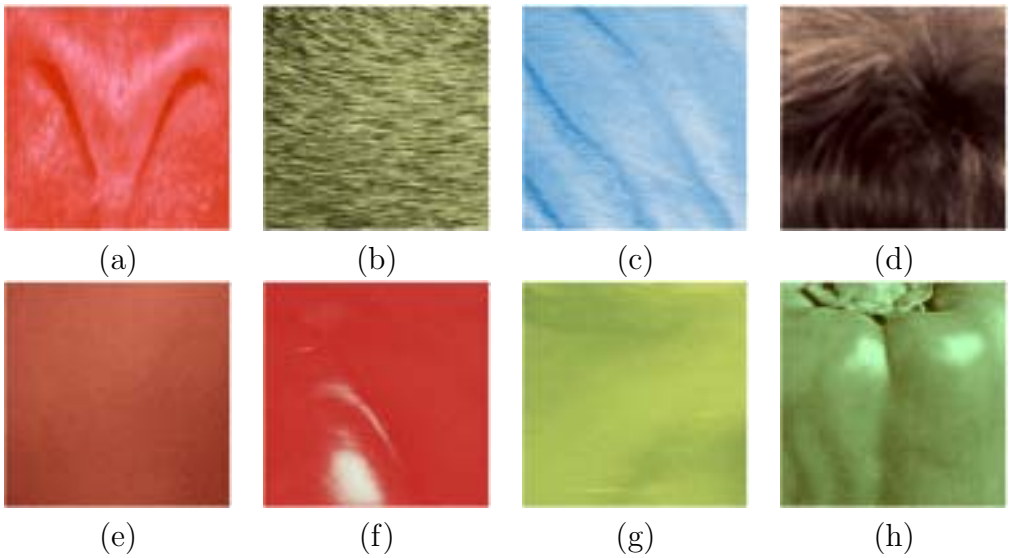


Figure 4.2: Results of the proposed dimension reduction process applied on the samples shown in Figure 4.1 (images are not proportionally resized.).

Table 4.1: Numerical results of the proposed dimension reduction method.

Swatch	**PSNR (dB)**	κ_{c_1}	κ_{c_2}	κ_{c_3}	κ_{pc_1}	κ_{pc_2}	κ_{pc_3}
(a)	25	0.28	0.20	0.52	0.72	0.26	0.02
(b)	30	0.34	0.38	0.28	0.96	0.04	0.00
(c)	30	0.63	0.22	0.15	0.80	0.20	0.00
(d)	35	0.42	0.32	0.23	0.99	0.01	0.00
(e)	38	0.41	0.31	0.28	0.95	0.05	0.00
(f)	34	0.04	0.51	0.45	0.97	0.03	0.00
(g)	33	0.46	0.29	0.25	0.78	0.20	0.02
(h)	28	0.18	0.42	0.40	0.89	0.08	0.03
η	32	0.35	0.33	0.32	0.88	0.11	0.01
σ	4	0.18	0.10	0.12	0.10	0.10	0.01

compared with 0.11 and 0.01). Comparison of κ_1 in the 1–st and the 4–th samples (the extreme values of κ_1) are noticeable too. While shades in the nostrils of the *Mandrill*, distributes the information in the three channels of PCA, the information densely focuses in pc_1 channel in the hair sample. This means that there is no relation between the spatial scattering pattern of the swatches and their energy compactions in the color domain.

The average PSNR value of $32dB$ in Table 4.1 shows that the proposed dimension reduction process is a reasonable solution for homogenous color vectors. Also, the low standard deviation of the PSNR values ($4dB$) proves that the method responds uniformly. The highest PSNR value is for the 5–th sample, the skin. The worst result is obtained in the 1–st sample because of the dark shades of the nostrils. Applying the proposed dimension reduction costs eight additions and four multiplications for each pixel. Investigating the results of the proposed dimension reduction process shown in Figure 4.2 proves its subjective performance. Thus, according to the typical results stated here and the numerous results of other sample images (not shown here due to length constrain), it is concluded that the proposed dimension reduction method performs reasonable for homogenous swatches of the color images of the nature.

4.2 Spotting

The sample image used in spotting test (see Section 3.2) were taken from 8 objects (stapler, infant, red ball, tin opener, spring, white ball, blue ball, and apple) with

Table 4.2: Spotting results in PLAC.

Sample	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Q_1	86.1	95.2	72.2	56.8	67.8	-	82.5	99.9
Q_2	100	0	100	100	100	0	100	100

different colors put on a smooth surface in the daylight by a digital camera (Figure 3.17).

The 41 channels of the 13 color spaces, PLAC, and PCA–PLAC, were investigated for the answers to the predefined questions. The results in the color spaces is too large to be printed in this thesis (to see the briefed version see Table 4.4 and Table 4.5). Tables 4.2 and 4.3 hold the results in PLAC, and PCA–PLAC, respectively. Tests on the 8 objects were performed 5 times for each object in PCA–PLAC and the average values of all δ_{ij} in different tests on the same object were computed as C along with the average value of all $\tilde{\delta_\lambda}$ as $\bar{\tilde{\delta}}_\lambda$ (Table 4.3). Table 4.4 and Table 4.5 compare fail rate, $\bar{Q_1}$, δ_Q and $\bar{Q_2}$ of the 41 channels, PLAC and PCA–PLAC.

Investigating the average and standard deviation of the spotting results in the standard channels is insightful. As shown in this table, the average value of Q_1 for most channels is higher than 50% but the standard deviation of the channels is too high (42.44%); which shows that the method is likely to act poor. It must be emphasized that in 73 tests the method is unable to find a reasonable portion of the object or to distinguish the border line, which leads to the desperate fail rate of 25%.

Over the stimuli the situation is even worse. The spotting method's success in

Table 4.3: Spotting results in PCA–PLAC.

Sample	$\bar{Q}_1$	δ_{Q_1}	$\bar{Q}_2$	$\tilde{\delta}_\lambda$	C
1	95	4.73	100	19	96.3
2	94.2	5.67	100	17	93
3	88	5.96	100	66	93
4	66.2	3.86	100	17	94.3
5	91.2	4.16	100	6	96.3
6	48.6	8.56	0	31	71.6
7	74.8	5.26	100	50	93.2
8	99.4	0.80	100	36	98
Average	82.17	4.88	87	30.25	92
Std. Dev.	16.78	2.19	30	19.36	8.94

color spaces entirely depends on the object. The best results have been recorded for the apple, the red ball, the spring, the blue ball and the stapler, which all make distinct loci in the color spaces. The worst result is captured for the white ball, because it is very similar to the background in color scheme.

In the $41 * 8 = 328$ attempts made for cutting the desired object out of the background, only 6 were successful to clear the entire area, which gives the poor mathematical expectation of less than 2%. This event is also very much depending on the subject, as 3 out of 6 has happened for the 7–th object.

Table 4.2 shows that in 6 out of 8 attempts, PLAC is successful to remove the entire background, which gives the promising result of 75%. This measure is 87% for PCA–PLAC as shown in table 4.3.

PLAC has failed to recognize the object in 12.5% of tests, which is half the fail

Table 4.4: Comparison of results in color spaces, PLAC and PCA–PLAC. (Part I)

Color Component	Fail	$\bar{Q}_1$	σ_{Q_1}	$\bar{Q}_2$
R	0	60.02	37.62	0
G	0	73.35	31.94	0
B	0	69.99	34.47	12.5
C	0	66.95	30.28	0
M	0	77.58	30.55	0
Y	0	71.84	33.05	0
K	37.5	42.83	39.18	0
$HSI.H$	37.5	52.27	48.79	12.5
$HSI.S$	12.5	77.81	33.19	0
$HSI.I$	25	55.07	43.12	0
$HSV.H$	37.5	55.29	48.23	12.5
$HSV.S$	25	58.50	48.96	0
$HSV.V$	25	47.45	38.48	0
I_1	25	55.93	46.65	0
I_2	50	46.49	50.44	12.5
I_3	37.5	57.42	47.91	0
$CIE-L$	25	60.94	44.01	0
$CIE-a$	37.5	60.97	50.50	0
$CIE-b$	37.5	58.90	48.98	0
$CIE-L$	25	60.94	44.01	0
$CIE-H$	37.5	61.80	51.19	0
$CIE-C$	25	65.54	42.65	0

Table 4.5: Comparison of results in color spaces, PLAC and PCA–PLAC. (Part II)

Color Component	**Fail**	Q_1	σ_{Q_1}	Q_2
$CIE - L$	25	60.94	44.01	0
$CIE - u$	62.5	34.52	48.06	0
$CIE - v$	37.5	57.62	48.45	0
$CIE - X$	25	58.92	44.58	0
$CIE - Y$	25	62.10	44.53	0
$CIE - Z$	12.5	69.97	37.57	0
$YC_bC_r.Y$	12.5	64.52	40.08	0
C_b	12.5	74.34	33.50	12.5
C_r	25	68.16	42.92	0
$YIQ.Y$	12.5	72.36	40.20	0
$YIQ.I$	37.5	48.62	45.83	0
Q	50	43.41	48.78	0
$YUV.Y$	12.5	70.78	41.27	0
$YUV.U$	12.5	84.43	34.41	0
$YUV.V$	50	49.28	52.69	0
$HMMD.H$	37.5	55.30	48.23	12.5
Min	25	58.51	48.96	0
Max	25	47.46	38.48	0
$HMMD.D$	25	77.45	33.09	0
Average	25	60.9	42.44	1.83
PCA–PLAC	0.00	82.18	4.88	87
PLAC	12.5	70.06	31.68	75

rate of the standard methods, having in mind that PCA–PLAC has never failed.

The expectation result of Q_1 in PLAC, is 70.06% with standard deviation of 31.68% in contrast with the average of 60.9% and standard deviation of 42.24% in the color spaces, showing about 10% better results with a smaller standard deviation when comparing PLAC to color spaces, making hope that PLAC responds uniformly in the stimuli range. Table 4.3 shows even better results for PCA–PLAC compared to PLAC. The surprising result of 82% expectation value with variance of less than 5% for Q_1 and 87% expectation for Q_2 when λ has changed about 30% shows the robustness of PCA–PLAC. It must be emphasized that the average correlation is more than 90%.

Also, as the two proposed PLACs are $1-D$ color descriptors, their computation time is at least three times less than ordinary $3-D$ color spaces. Compared to the sophisticated hue–saturation based color spaces, which use complicated functions, the performance of the PLACs are far more better. Further more, PLAC and PCA–PLAC are very much appropriate for analog implementation by ordinary circuitry.

The important disadvantage of PLAC is the tedious tuning job, which reduces its repeatability and needs supervision of human observer, a shortcoming that is removed in PCA–PLAC. It is easy to see that PCA–PLAC needs only one parameter to be tuned by the user in contrast with the two parameters in spaces and five parameters in PLAC. Also, user has no intuition when setting a_r, a_g, a_b, C, T parameters in PLAC

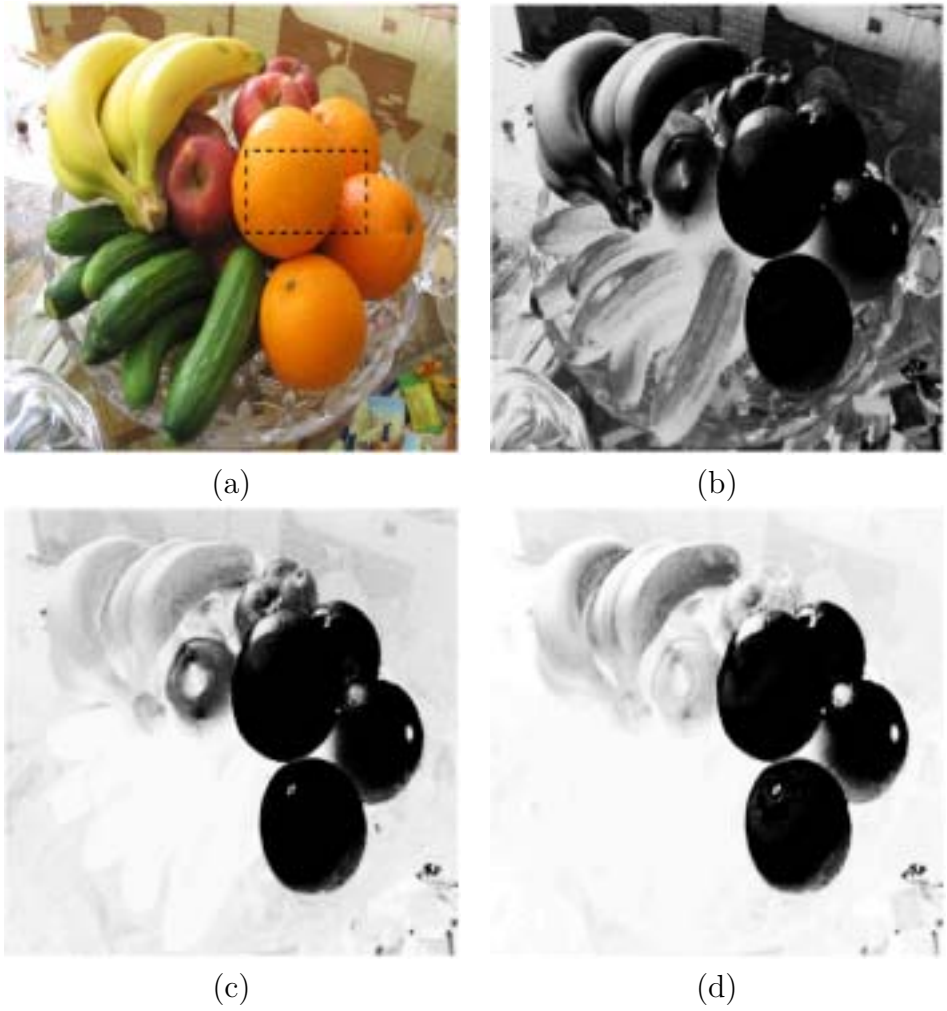
in contrast with the meaningful λ parameter in PCA–PLAC.

PCA–PLAC has appeared surprising to gain $Q_1 = 46.8\%$ for the peculiar 6–th object, where all other methods, even the PLAC have failed.

4.3 Distance Measures

The first evidence for suitability of this cylindrical modelling comes from using a set of models on a database of homogenous swatches and interpreting it's statistics. Figure 4.3 shows a typical comparison between different distances for image fuzzification. Here, the user gives a query region and the algorithm fuzzifies all pixels according to their relevance to the color content of the given region. Figure 4.3–a illustrates a sample image and the query region located on the orange. Figures 4.3–b, c, and d show the results of the Euclidean, Mahalanobis, and the LPRE distances, respectively. Here, black color denotes the maximum membership while white denotes complete irrelevance. Although the Euclidean distance has ranked almost all points corresponding to the orange with high membership values, there are lots of other points in the apple, banana, and even cucumber that have gotten hight values of membership. The Mahalanobis distance is giving more promising results, however, with small portions of the apple being highlighted too. The LPRE performs the best in this test.

Another comparison method between different distance functions is the query region extraction task; which is related to the performance of a given distance function



(a) (b)

(c) (d)

Figure 4.3: Performance comparison of the distance functions in fuzzifying a given color image according to a query region. (a) Original image adopted from [318] along with the query region, (b) Euclidean. (c) Mahalanobis. (d) LPRE.

in extracting the region related to the given query region (see [89] for details about query region extraction). Figure 4.4–a shows a sample image and the given query region. Figures 4.4–b, c, and d show the results of query region extraction using all three mentioned distances. The outperforming tendency of the LPRE distance is clear. Note that in contrast with the fuzzification task, in which Mahalanobis is giving acceptable results, here it is performing desperately. The reason behind this twofold habit is the closed–shape of the Mahalanobis distance's iso–curves while LPRE curves are non–crossing parallel lines. Hence, the Mahalanobis distance is acceptable in rejecting irrelevant points but it also rejects many relevant points.

The third method to compare different distance functions is to use them as homogeneity criterion and then interpret the differences. As mentioned above, each distance function produces a set of iso–curves. Thus, for each given swatch, one can decide on its homogeneity by finding the iso–curve occupied averagely by the points of the swatch. Note that in this way one finds the average distance of a swatch's members to itself. Figure 4.5–a shows a sample image, out of which 49 swatches are randomly selected. This swatches are sorted according to the homogeneity factor given by the three mentioned distances. Figures 4.5–b, c, and d show the obtained results. In each set of regions, the top left image is known to be the most homogeneous region, while the bottom right region is believed to have the least homogeneity. Investigating the order of the regions computed by the Euclidean distance (Figure 4.5–b) shows that

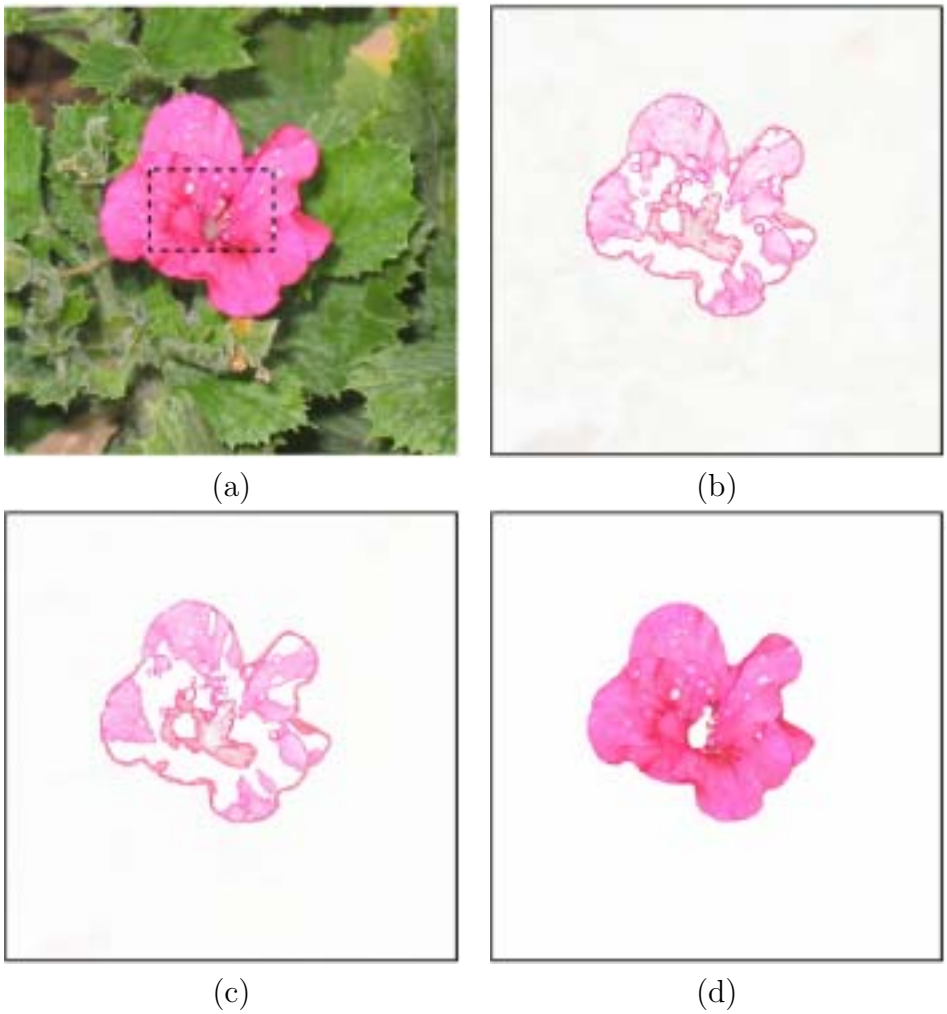


Figure 4.4: Performance comparison of the distance functions in query region extraction. (a) Original image along with the query region, (b) Euclidean. (c) Mahalanobis. (d) LPRE.

the ordering does not comply with the human perception. For example compare the regions 40 to 43 with the 37–th and the 38–th one. Although, the Euclidean distance shows that the 37–th and 38–th regions are more homogeneous, human observers order them inversely. The same contradiction happens in the set of regions ordered by the Mahalanobis distance (Figure 4.5–c). For example, compare the 15–th and the 14–th or the 28–th and the 4–th regions. Investigating Figure 4.5–d shows the similarity of the human perception and the LPRE homogeneity criteria. Looking at the set of the regions ordered by the LPRE, from the first to the last shows that the first regions ($1 \cdots 24$–th) contain almost just one dominant color, while the regions of two and three colors come afterwards. The fourth cue for comparing different distance functions is to manually decompose an image into homogenous regions and then to investigate the loci of the points in the same group. In this way one can judge the proper cluster shape (ruling the proper distance function).

4.4 Query Region Extraction

4.4.1 LPRE–Based Approach

Figure 4.6 shows the sample images collocated with the corresponding query regions. The desired result of query segmenting each sample image using the corresponding query regions, is simulated manually (see Figure 4.7). These masks are used as the ground truths. Note that to give a better printing result, the backgrounds are shown in white, while for the points in the mask, the corresponding pixels in the original

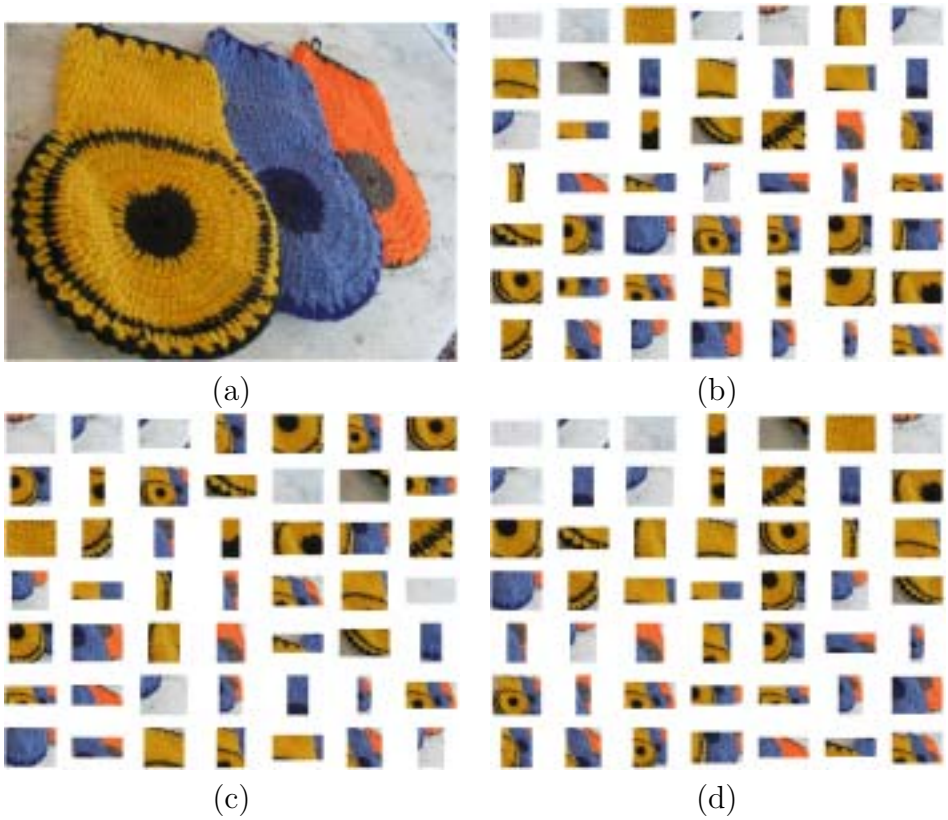


Figure 4.5: Performance comparison of the distance functions in homogeneity decision. (a) Sample image adopted from [318]. (b), (c), and (d) Randomly selected regions sorted in terms of different homogeneity criteria in a descending fashion (b) Euclidean. (c) Mahalanobis. (d) LPRE.

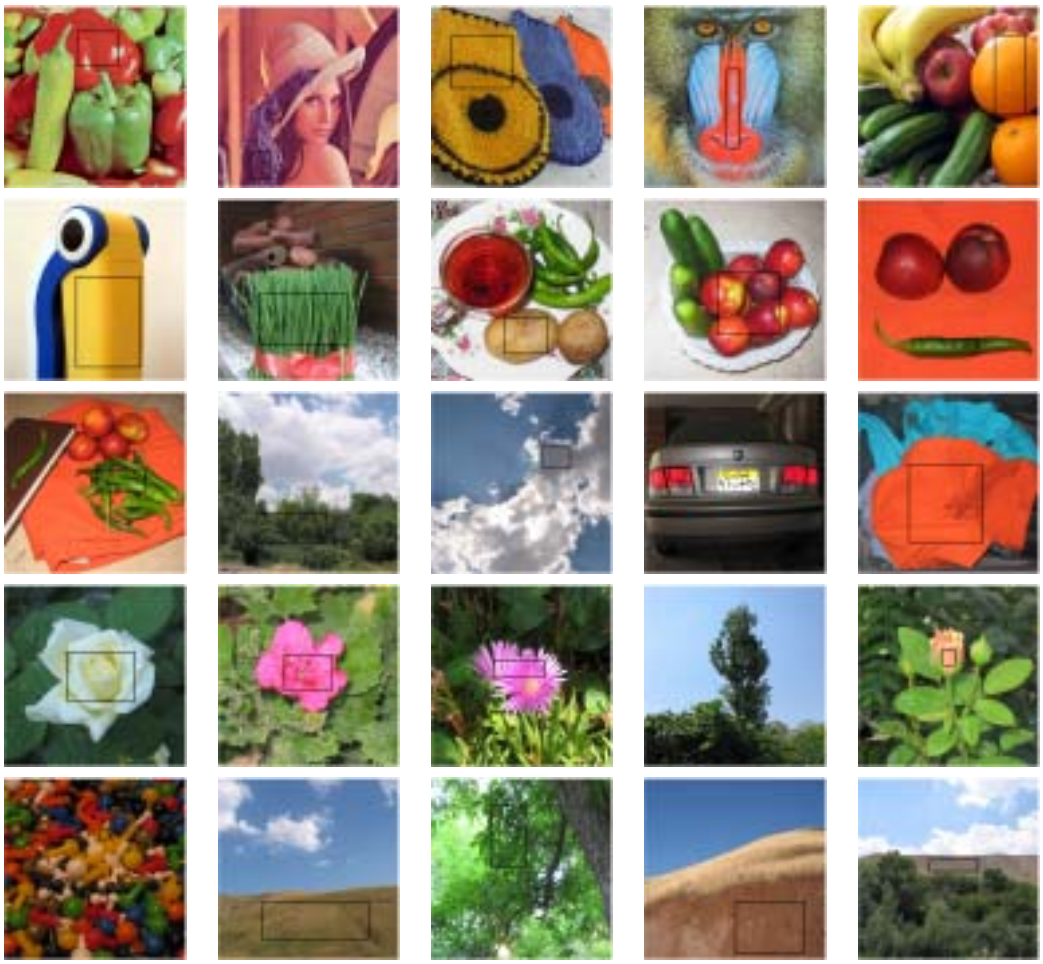


Figure 4.6: Sample images and query regions.

image are used.

For each sample image, the proposed query region extraction is performed $12 \times 7 + 2 = 86$ times. This number comes from 7 Euclidean distances corresponding to the 7 representations of each of the 12 color spaces plus the Mahalanobis and the LPRE distances. The 7 representations of a color space, when used in RGB, result in spaces made by R, G, B, RG, GB, BR, and RGB, as an example. In each representation, only the selected components are used in measuring the Euclidean distance. Note that the only parameters of the proposed query region extraction, are the radius ρ and the inclusion percentage p. While ρ controls the spatial smoothness of the resulting mask, p serves as a margin influencing the result massively. Thus, in this stage, ρ is set constantly equal to 5. Also, p is set to 0.75 and the $86 \times 25 = 2150$ query region extraction tasks are performed and the corresponding values of μ_r and μ_p are computed. Tables 4.6 and 4.7 shows the results on different images. Selecting those likelihood measures with both recall and precision rates of more than 0.7, the only choice is LPRE. Note the promising results of the LPRE, compared to all others, even to the Mahalanobis distance. The same results are observed for different values of p. Thus, LPRE is the best available solution for query region extraction. In the rest of this thesis, the LPRE is used.

Another test is performed to investigate the effects of p on μ_r and μ_p. All samples images were analyzed by the proposed query region extraction, while setting $\rho = 5$

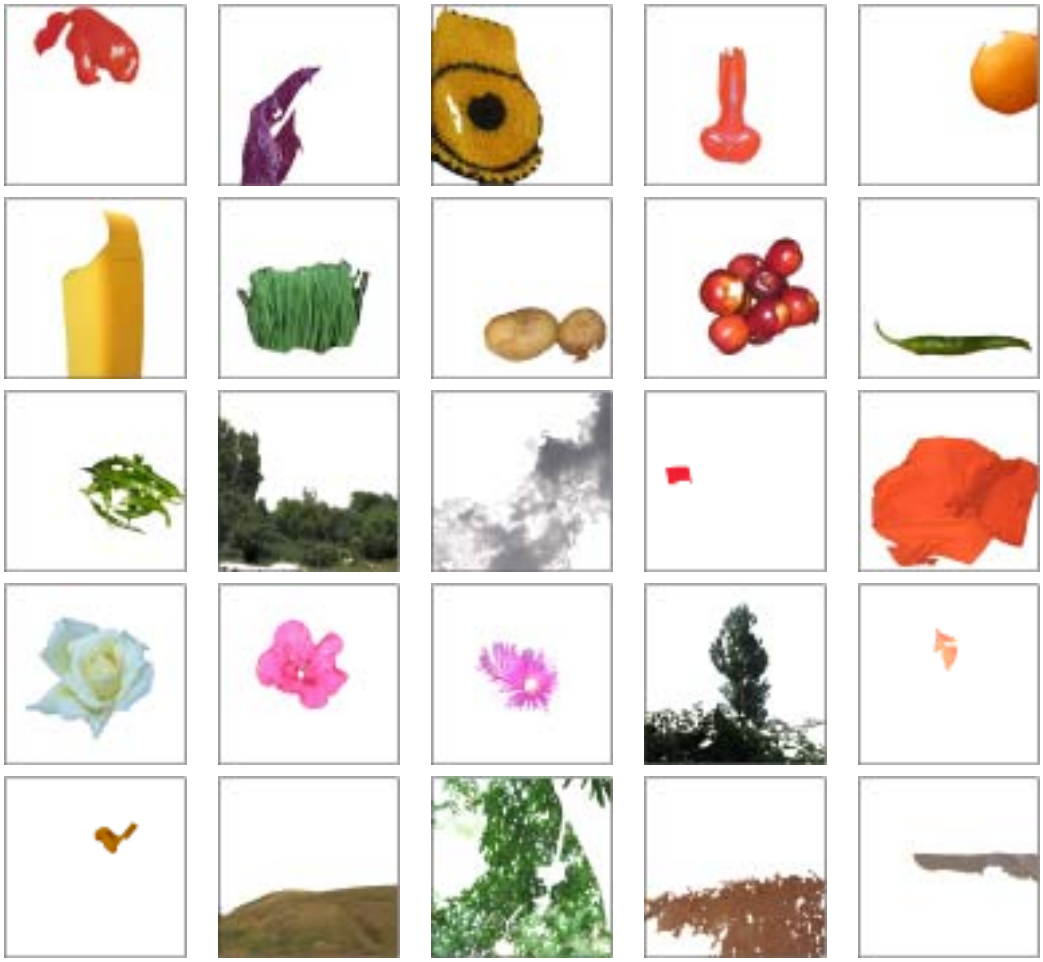


Figure 4.7: Ground–truth results of query region extraction for the images shown in Figure 4.6 with corresponding query regions.

Table 4.6: Query region extraction results using the Euclidean distance in twelve color spaces, Mahalanobis, and LPRE. Part I [μ_r: recal, μ_p: precision]

Color Space	1–D					
	c_1		c_2		c_3	
	μ_r	μ_p	μ_r	μ_p	μ_r	μ_p
RGB	0.57	0.82	0.61	0.72	0.62	0.79
$HMMD$	0.61	0.87	0.64	0.77	0.58	0.80
$I_1I_2I_3$	0.59	0.87	0.60	0.84	0.63	0.87
HSI	0.61	0.87	0.64	0.65	0.59	0.87
$CIE-XYZ$	0.58	0.87	0.60	0.80	0.63	0.78
HSV	0.54	0.76	0.63	0.70	0.58	0.80
YC_bC_r	0.60	0.80	0.62	0.86	0.61	0.94
$CIE-Lab$	0.61	0.80	0.68	0.84	0.66	0.78
YIQ	0.60	0.80	0.60	0.92	0.64	0.87
$CIE-LHC$	0.61	0.80	0.60	0.81	0.68	0.72
YUV	0.60	0.80	0.62	0.86	0.60	0.94
$CIE-Luv$	0.61	0.80	0.63	0.90	0.67	0.81

Table 4.7: Query region extraction results using the Euclidean distance in twelve color spaces, Mahalanobis, and LPRE. Part II [μ_r: recal, μ_p: precision]

Color Space	2–D						3–D	
	c_1,c_2		c_2,c_3		c_3,c_1			
	μ_r	μ_p	μ_r	μ_p	μ_r	μ_p	μ_r	μ_p
RGB	0.57	0.96	0.59	0.84	0.55	0.93	0.56	0.99
$HMMD$	0.60	0.95	0.57	0.96	0.56	0.94	0.55	0.99
$I_1I_2I_3$	0.57	0.92	0.57	0.96	0.57	0.97	0.56	0.99
HSI	0.63	0.91	0.59	0.90	0.57	0.99	0.57	0.99
$CIE-XYZ$	0.59	0.88	0.59	0.90	0.56	0.92	0.57	0.94
HSV	0.55	0.82	0.57	0.97	0.52	0.83	0.53	0.92
YC_bC_r	0.59	0.90	0.57	0.97	0.58	0.96	0.57	0.99
$CIE-Lab$	0.60	0.85	0.62	0.91	0.59	0.90	0.58	0.92
YIQ	0.58	0.94	0.58	0.97	0.59	0.94	0.57	0.99
$CIE-LHC$	0.54	0.89	0.58	0.87	0.58	0.89	0.53	0.88
YUV	0.59	0.90	0.57	0.96	0.58	0.96	0.57	0.99
$CIE-Luv$	0.57	0.97	0.60	0.98	0.58	0.93	0.56	1.00
Mahalanobis							0.52	0.99
LPRE							0.92	0.97

with varying values of p in the interval $[0.05, 0.95]$. Figure 4.8 shows the average recall and precision rates. As expected, the recall rate grows as p converges to the unity. Also, the precision grows and then declines by increasing p, because in the first phase we are adding relevant points and in the second phase we are adding irrelevant ones to the color content. Note that the midway, $p = 0.5$, is a good compromise between μ_r and μ_p, resulting in μ_r of about 0.9 and μ_p of more than 0.98. It worths to mention that the average elapsed time is 530ms with the standard deviation of 49ms with no explicit dependency to p. Figure 4.9 shows the same measures while setting $p = 0.5$ constantly and varying ρ. Figure 4.10 shows the corresponding elapsed times. As Figure 4.9 shows, the precision is not changing with ρ significantly. One decade increase in ρ, produces a 0.1 declines in the recall rate. The effect of ρ on recall rate, relates to the over–smoothed edges. Figure 4.10 shows that the elapsed time relates quadratically to ρ, because of the increased computing demands of the averaging convolution kernel. In all cases the average elapsed time is less than 800ms.

Figures 4.11 illustrate the visual results of changing p and ρ. As expected, increasing p too much, vanishes the performance of the resulted segment by widening the color content interval excessively, while large ρ over–smoothes the segment, forcing to ignore the details. These results are complying with the curves shown in Figure 4.9.

Figure 4.12 shows a sample results of the repeatability test. Here, 10 distinct query regions containing the same material, are manually selected and the proposed

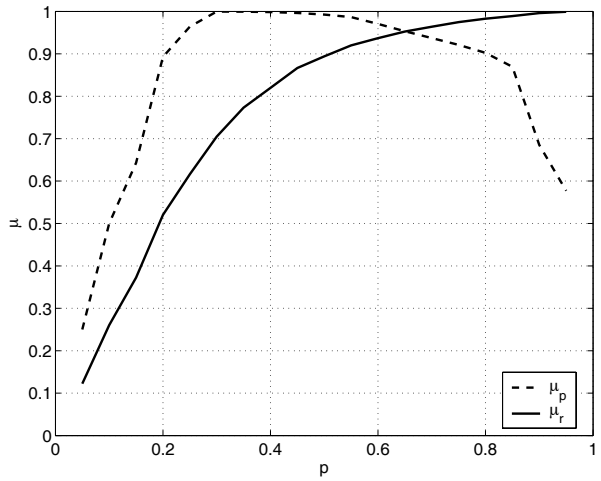


Figure 4.8: Average precision and recall rates of the proposed query region extraction method using LPRE, with varying p.

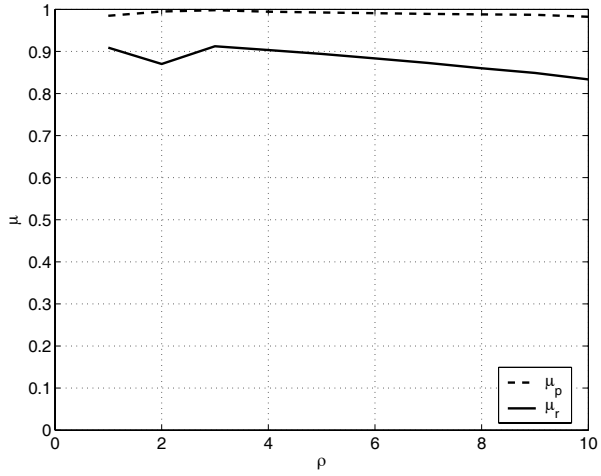


Figure 4.9: Average precision and recall rates of the proposed query region extraction method using LPRE, with varying ρ.

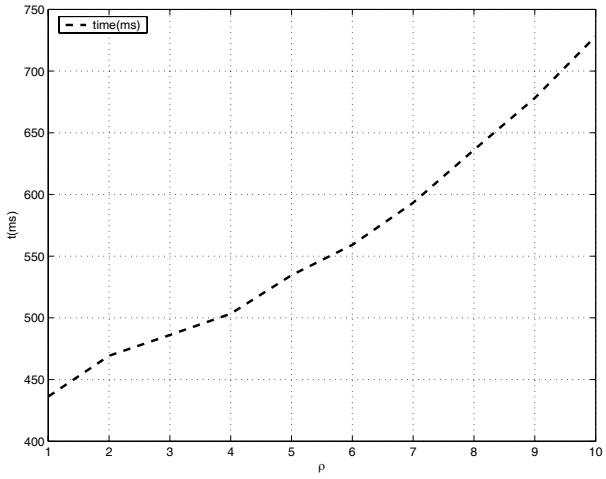


Figure 4.10: Elapsed time of the proposed query region extraction using LPRE, with varying ρ.

query region extraction is performed on them. Analysis of the 90 values of correlation between the resulting masks, shows the mathematical expectation of more than 0.98 with standard deviation of less than 0.02. Along with other observations, this proves the repeatability of the proposed method.

Figure 4.13 shows a typical result of the robustness test of the proposed method to the existence of the outliers. Note that while using partially non–homogenous regions as the query region, the proposed method shows good levels of robustness towards rejecting the irrelevant vectors. In this test, using 5 different regions containing averagely 5% irrelevant points, the average correlation of the results is 0.97% with standard deviation of 5%.

(a)

(b)

Figure 4.11: (a) Effects of changing p in query region extraction. (a) Effects of changing ρ in query region extraction.

Figure 4.12: Typical results of the repeatability test.

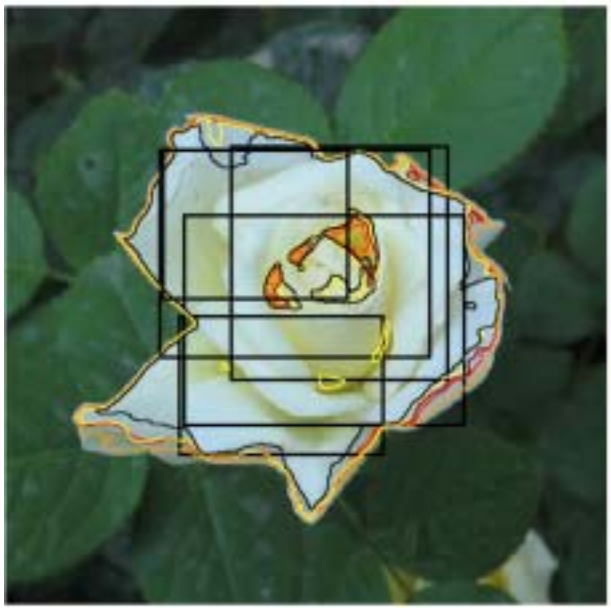


Figure 4.13: Typical results of the robustness test.

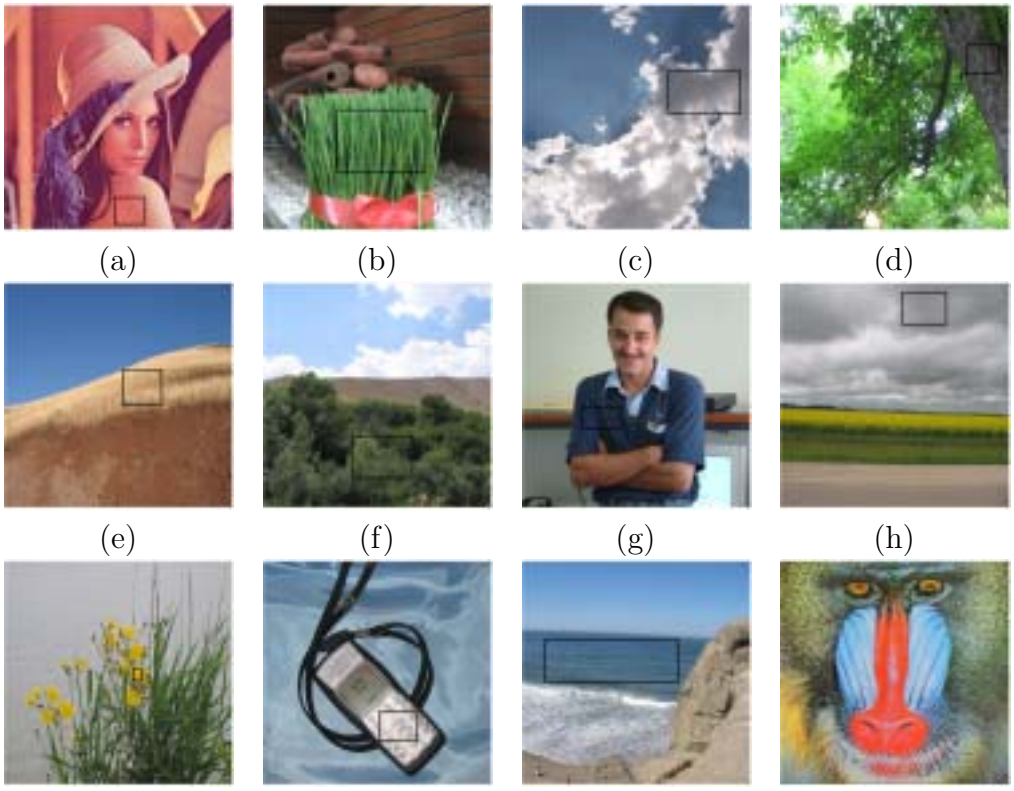


Figure 4.14: Samples images and corresponding query regions.

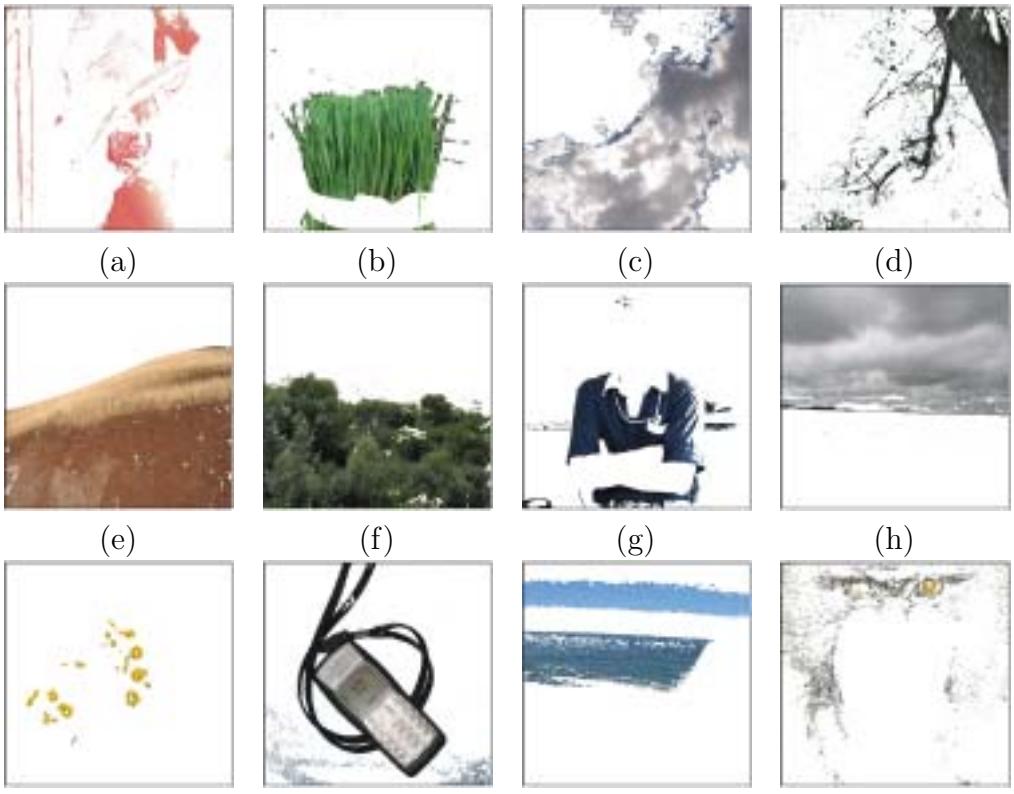


Figure 4.15: Query region extraction using the LPRE.

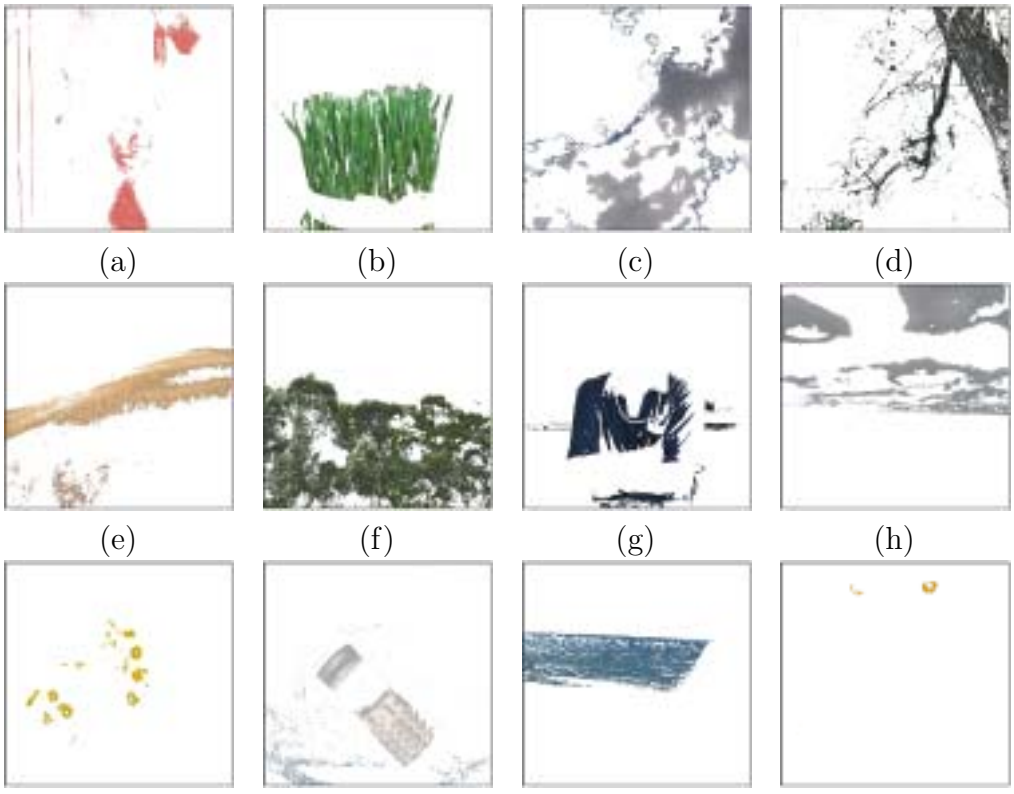


Figure 4.16: Query region extraction using the proposed LLPRE.

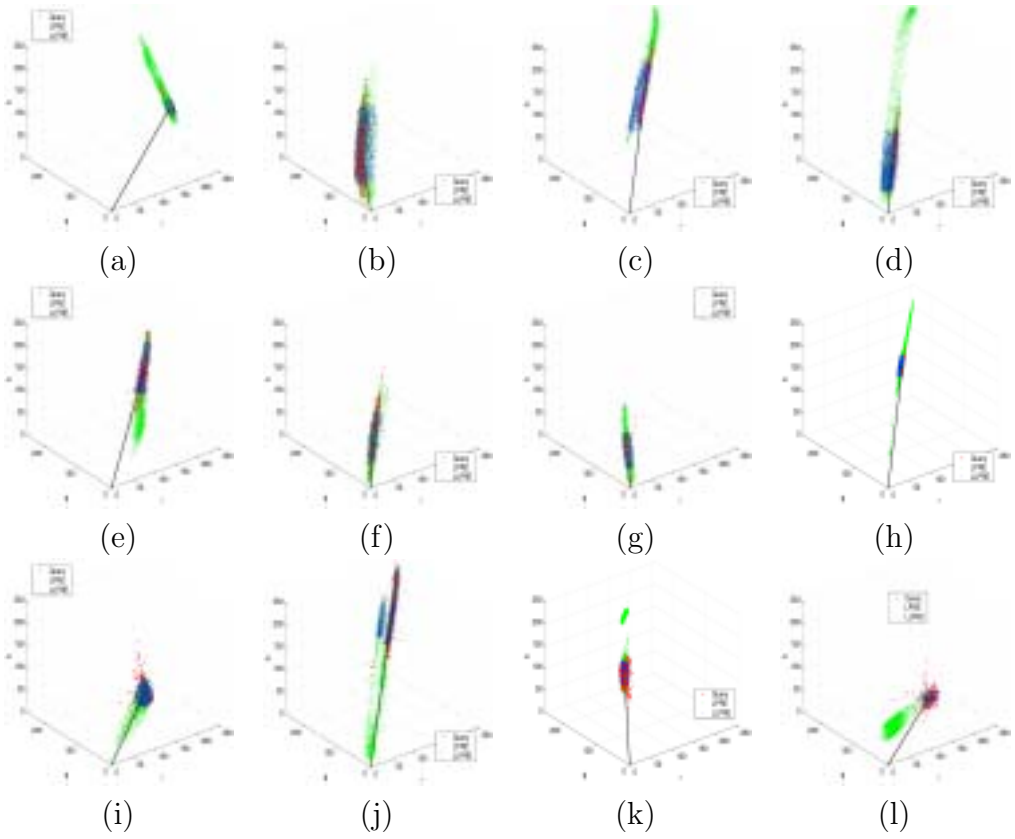


Figure 4.17: Color vectors of the query region, the result of LPRE (shown in Figure 4.15), and the result of LLPRE (shown in Figure 4.16).

4.4.2 LLPRE–based Approach

Figure 4.14 shows the sample images used for analysis of the performance of the proposed LLPRE distance function. In each case the corresponding query region is also illustrated. Figure 4.15 and Figure 4.16 show the results of query region extraction using LPRE and the proposed LLPRE, respectively. Note that in all cases LLPRE has resulted in a narrower selection of the color vectors, and that always LPRE contains the results of LLPRE. Investigating the set of points omitted by LLPRE while being accepted by LPRE is promising. Figure 4.15–b given by LPRE contains the dark shades of the grass and some other black parts of the wall. None of these points are contained in Figure 4.16–b which is produced by LLPRE. Note that in Figure 4.14–b the query regions mostly contains the grass points and few dark shades really exist in it. The same happens for the bright clouds in Figure 4.15–c and Figure 4.15–h, sky in Figure 4.15–d, soil in Figure 4.15–e, and dark shades in Figure 4.15–f, Figure 4.15–g, and Figure 4.15–i. Also, in Figure 4.15–j the dark body of the phone is included while it does not exist in the query region at all. These are all results of misclassification performed by LPRE. Note that in Figure 4.16–j LLPRE performs desirably. In Figure 4.15–k and Figure 4.15–l, LPRE has mistakenly included the bright sky and the dark wool. Figure 4.17 reveals the reason behind the differences between the results of LPRE and LLPRE. In each image, the color vectors belonging to the query region are drawn in red, while the result of LPRE and LLPRE

are illustrated in green and brown, respectively. Note that as LPRE models clusters as infinite cylinders, there is a possibility for inclusion of the (partly) irrelevant color vectors that lie in the far end of the cluster. Investigating Figure 4.17 proves that most of the times this the reason for the erratic activity.

In the discussed samples, LPRE elapses averagely $300ms$ while LLPRE takes $500ms$ in average showing about 80% inflation of the processing demand. This is related to the computation of the second term in the proposed LLPRE distance.

4.5 Alpha Estimation

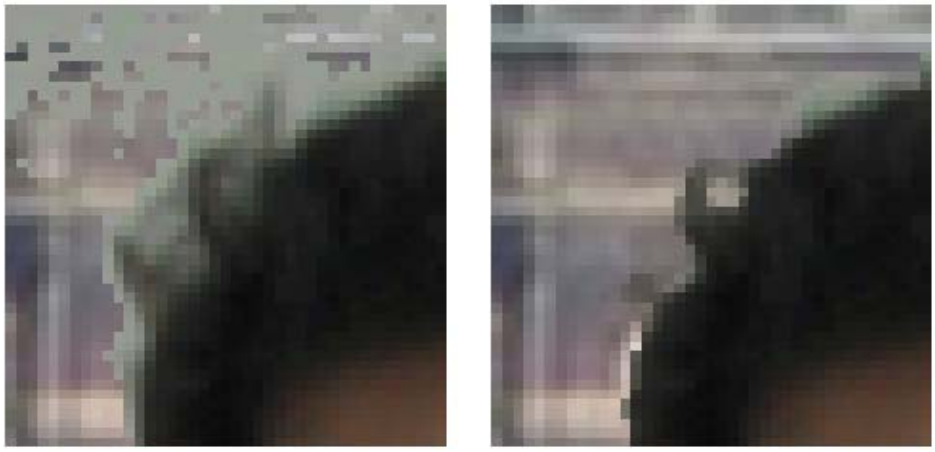
Figure 4.18 show the performance of utilizing the proposed query region extraction method for Alpha estimation. Comparing figures 4.18-c and 4.18-d makes clear that while the canonical Euclidean method is unable to compensate the narrow range of changes in the sample background points (see figure 4.18-a) it results in holes in the main object. More than better classification of background and object by the proposed method, it also preserves the sub–pixel edges and fuzziness of them. The comparison of the results is more clear in figure 4.19 which shows a magnified portion of both results.

(a) (b)

(c)

(d)

Figure 4.18: Result of the proposed query region extraction for alpha estimation. (a) Original image used as the background adopted from [2]. (b) Foreground image. (c) Canonical method. (d) Proposed method.



(a) (b)

Figure 4.19: Magnified portions of the the montage images shown in Figure 4.18, (a) Canonical method. (b) Proposed method.

4.6 Segmentation

4.6.1 FPCA–based Clustering (FPCAC)

Figure 4.20 shows some of the test images. Figure 4.21 shows one of the tests performed for cluster shape investigation purpose. Four base regions, corresponding to distinct materials (grass, brick, ribbon, and sand) are selected. Figure 4.22 shows the resulting clusters. Figure 4.23 depicts the computed clusters in different color spaces. Note that in the experiments, the dynamic range of all three components of each color space are stretched to fit the range $[0, 255]$, using linear mapping. Investigating Figure 4.23 shows that the assumption of spherical or ellipsoidal clusters is not satisfied in none of the tested color spaces. In fact some color spaces, (including $HMMD$, HSI, $CIE-Luv$, $CIE-Lab$, and $CIE-LHC$), deform the clusters to

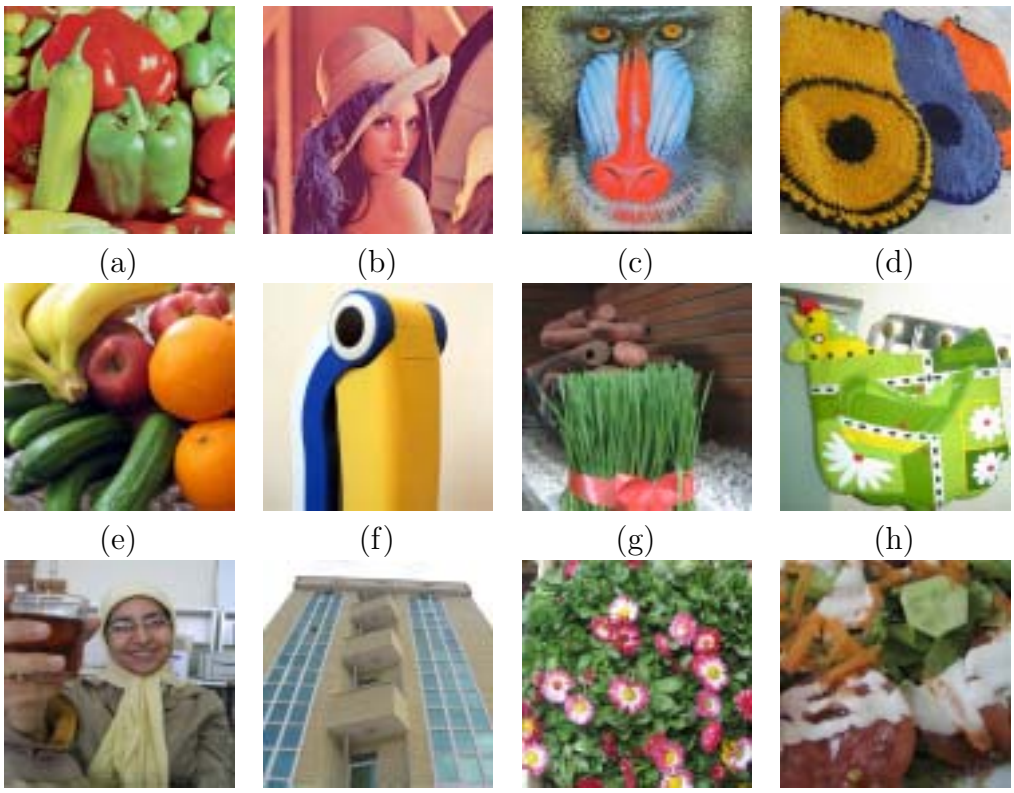


Figure 4.20: Some of the test images, (a)-(c) standard images, (d)-(l) adopted from [318].

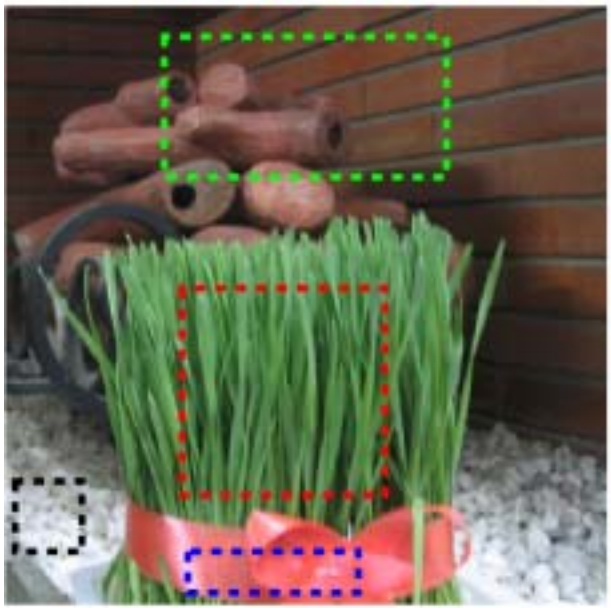


Figure 4.21: Sample image and four base regions.

non–similar shapes, (for example see the scattered clusters in the $CIE - Lab$ color space). On the other hand, the HSV splits the clusters into non–adjacent parts, (because of the 2π discontinuity of the trigonometric functions). The six other color spaces are those that are linearly related to the RGB color space [166]. Note that the overall shape of the clusters is not changed in these spaces, while they have increased the computation cost. The same results were obtained when running on other test images. Consequently, as the results show in none of the color spaces under investigation, the shape of the clusters can be reasonable appropriated by the spherical or the ellipsoidal clustering methods.

Figure 4.24–b shows the results of applying the proposed clustering method on an

Figure 4.22: Clusters corresponding to the base regions shown in Figure 4.21.

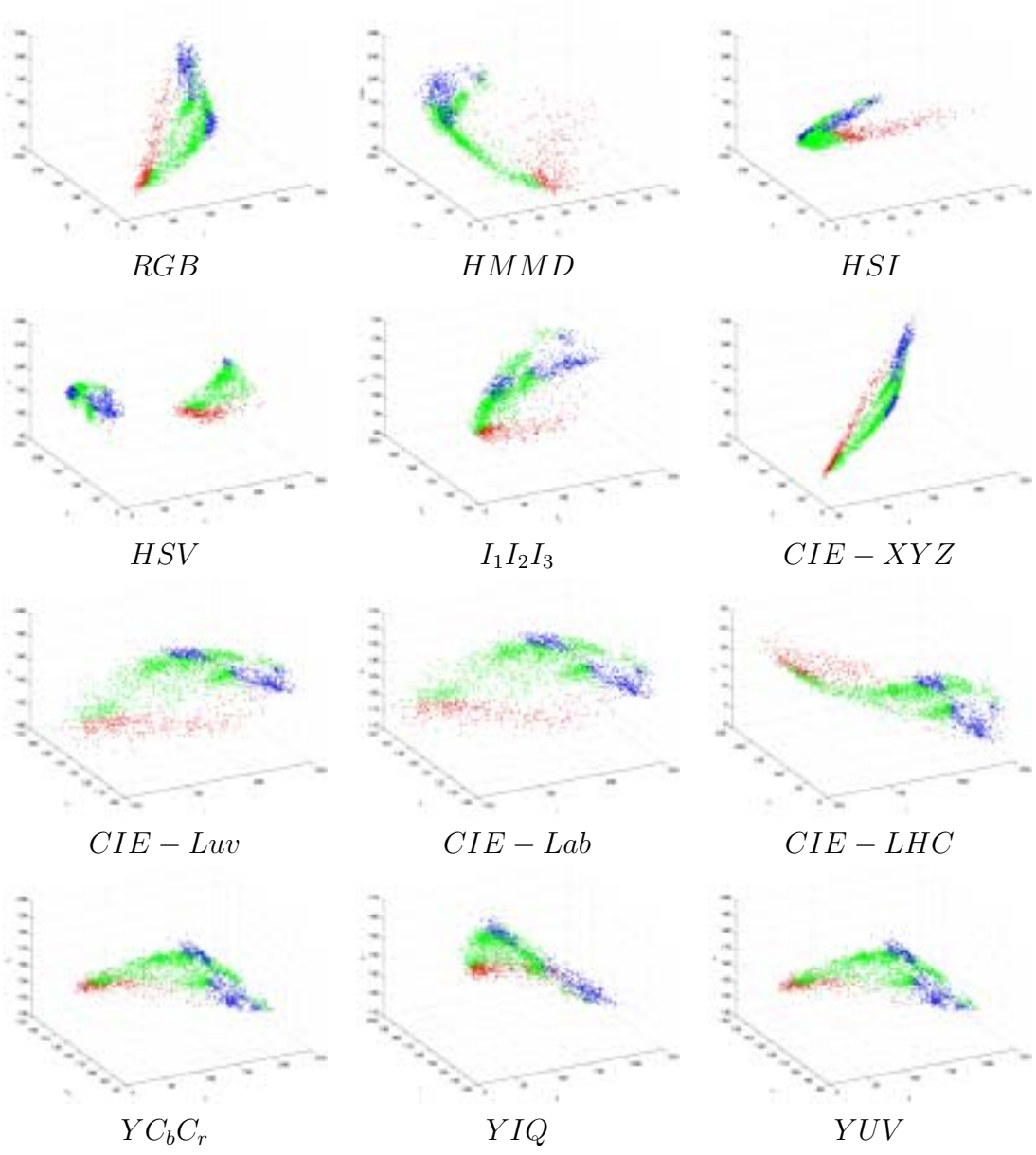


Figure 4.23: Shapes of the clusters belonging to the material highlighted in Figure 4.21 in different color spaces.

artificial data shown in Figure 4.24–a. The parameters are set as $c = 3$, $m = 1.05$, and $\varepsilon = 0.1$. The method converged in 12 iterations. The magenta lines in Figure 4.24 show the axis of clusters, while the red crosses show the center of the computed clusters. The colored curves illustrate the path of the cluster centers towards the convergence state.

Figure 4.25 shows the results of applying the proposed clustering method on the image shown in Figure 4.20–a, using different numbers of clusters. In this experiment, the parameters were selected as $m = 1.05$ and $\varepsilon = 0.5$. The method converged in 3, 12, 18, and 18 iterations, for consecutive values of $c = 2 \cdots 5$. Figure 4.26 also shows the results when using the image shown in Figure 4.20–e. Here, the method converged in 8, 10, 16, and 25 iterations, respectively. Investigating Figures 4.25 and 4.26 shows that selecting small values of c results in merging clusters in a sensible manner. For example in Figure 4.25 the method has resulted in the clusters of {red,green} and {light green, red, dark green} for the values of $c = 2$ and $c = 3$, respectively. In Figure 4.26 the clusters are {hot, cold}, {yellow, green, reddish}, and {yellowish, green, red, dark} for the choices of $c = 2$, $c = 3$, and $c = 4$, respectively. Although, when trying to decompose an image into large number of clusters, (big c), over clustering takes place, the resulting clusters are not spurious. Note that in all tests, (on the proposed method and on the FCM), to reduce the elapsed time of the algorithms, the number of the pixels are reduced by a factor of 20 using a sampling method in the

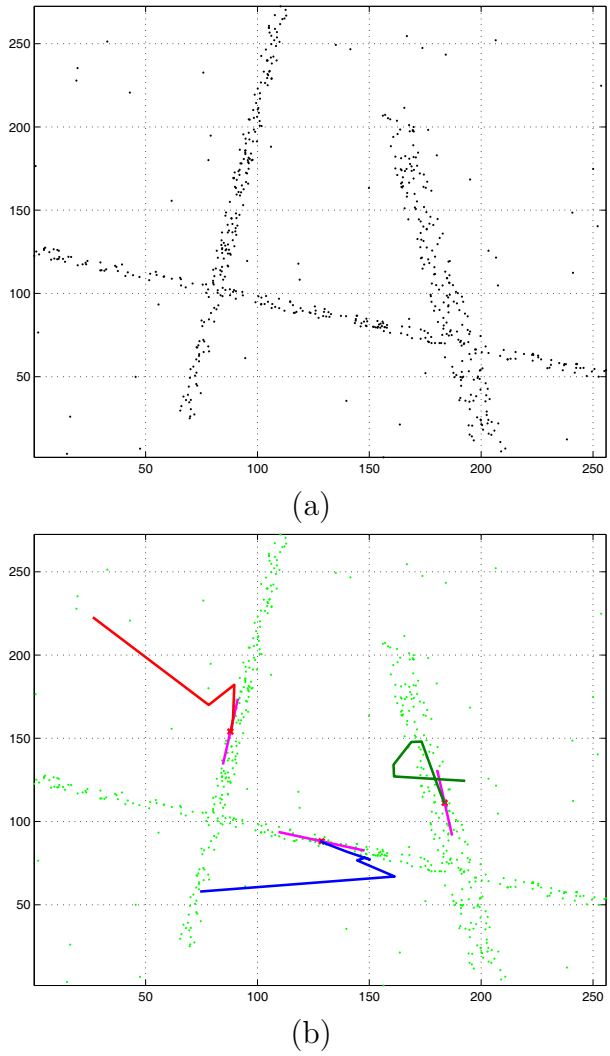


Figure 4.24: Results of applying the proposed clustering method on an artificial data. (a) Data. (b) Results.

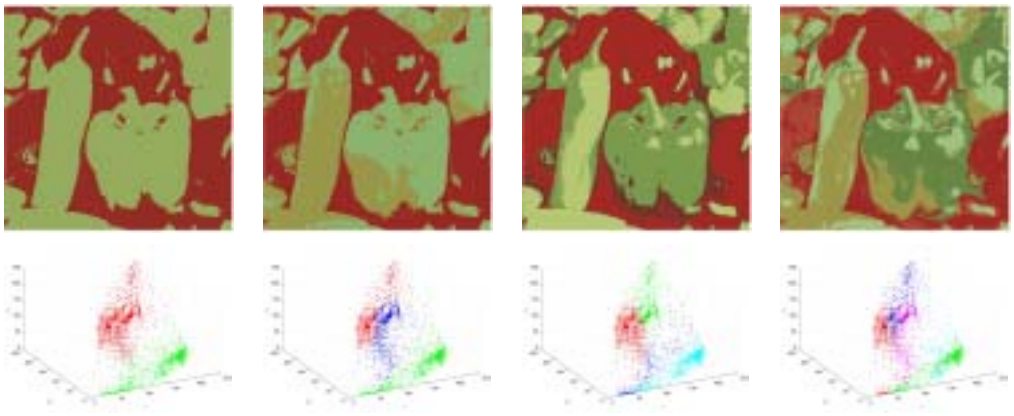


Figure 4.25: Results of applying the proposed clustering method on image shown in Figure 4.20–a when using different number of clusters.

rows.

Note that the only source of the uncertainty in the proposed clustering method is the random initialization stage. To measure the repeatability of the proposed clustering method, the image shown in Figure 4.20–b is clustered into 2 clusters with 25 different initialization. Here, the parameters were set as $m = 1.05$ and $\varepsilon = 0.5$. Figure 4.27 shows some results. The case of Figure 4.27–c is shown to visualize the other probable clusterings. The similarity of two maps i and j is defined as,

$$\rho_{i,j} = \frac{E\{ij\}}{\min\left\{E\{i^2\}, E\{j^2\}\right\}}. \tag{4.6.1}$$

To consider the case of cluster replacement, the similarity between different clusterings of a single image into two sets of $I = \{i_1, i_2\}$ and $J = \{j_1, j_2\}$, is defined as,

$$\rho_{I,J} = \max\{\frac{\rho_{i_1,j_1} + \rho_{i_2,j_2}}{2}, \frac{\rho_{i_1,j_2} + \rho_{i_2,j_1}}{2}\}. \tag{4.6.2}$$

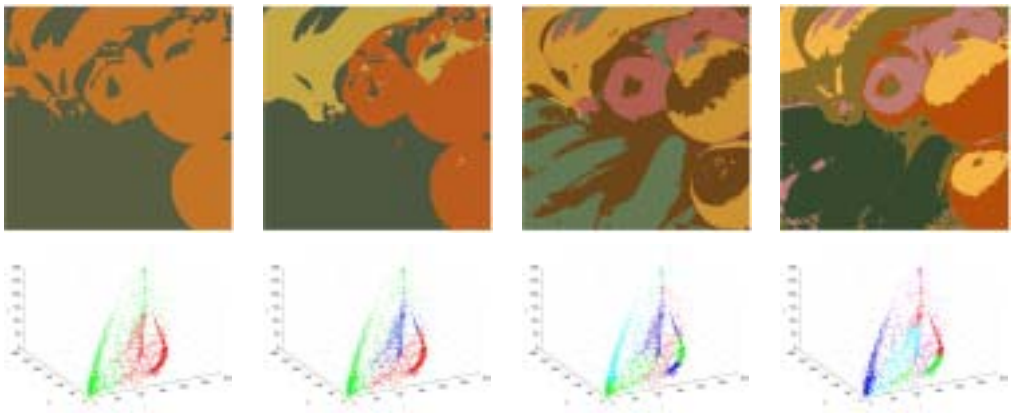


Figure 4.26: Results of applying the proposed clustering method on image shown in Figure 4.20–e when using different number of clusters.

Thus, for n times repeating the clustering algorithm on the same image with different initializations, $\frac{1}{2}n(n-1)$ values of correlation are measured. For the typical image, (Figure 4.20–b), the average and the standard deviation of the correlation results were measured as 0.83 and 0.21, respectively. It must be emphasized that the minimum and maximum correlation results were equal to 0.52 and 1.00, respectively and about 60% of the results were more than 0.99. Figure 4.28 shows the histogram of the correlation values. This almost two–modal pattern has occurred because of the two possible clusterings of the given image (Figures 4.27–c and 4.27–a). The same results are obtained on other test images.

To investigate the rule of the fuzziness parameter, m, the image shown in Figure 4.20–h is used. The proposed clustering method is performed on the image, using

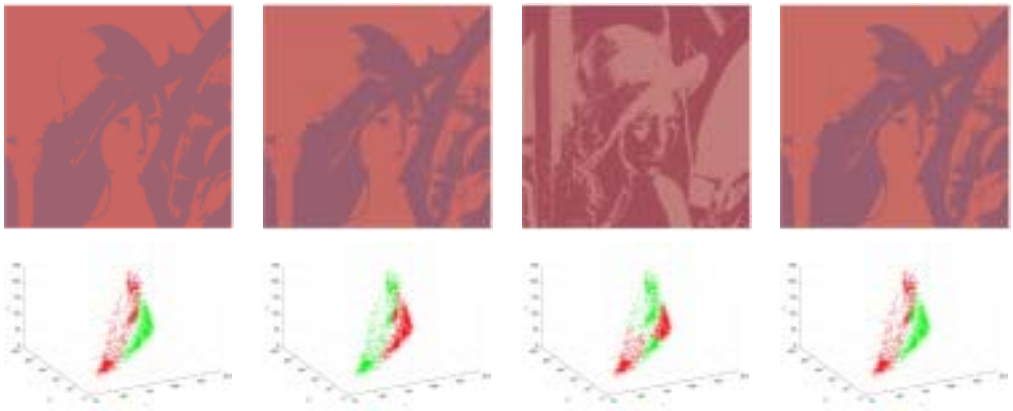


Figure 4.27: Repeatability test on the image shown in Figure 4.20–b.

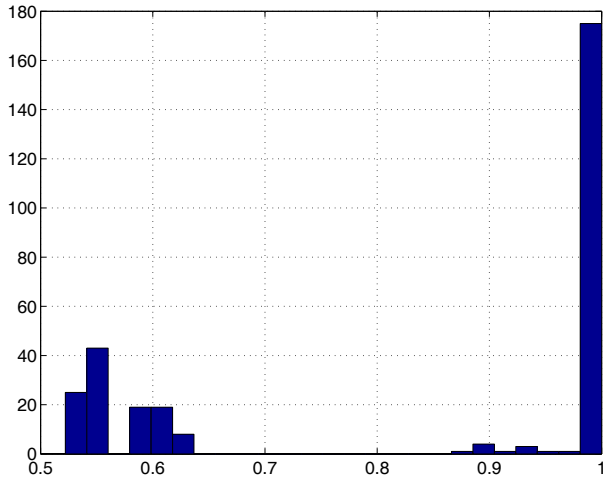


Figure 4.28: Histogram of the correlation of clustering results performed on the same image.

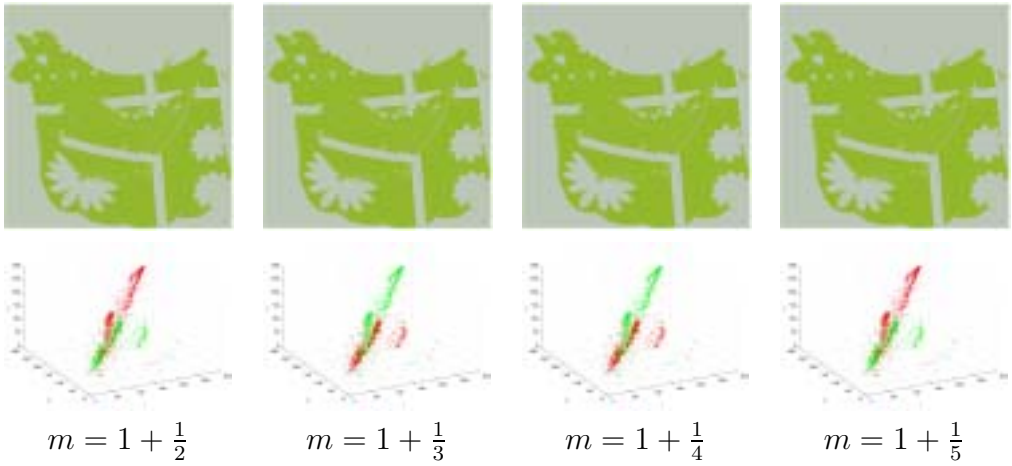


Figure 4.29: Results of different fuzziness values on the clusters corresponding to the image shown in Figure 4.20–h.

the parameters set as $c = 2$, $\varepsilon = 0.5$ and $m = 1.5, 1.33, 1.25, 1.2$. The results are shown Figure 4.29 for the results. Note that different values of m did not have serious effect on the final clustering results in this special example. Figure 4.30 shows the case that the value of m do affect the cluster shapes.

To test the computational cost of the proposed clustering method, the images were processed using different values of m. Figure 4.31 shows the time performance and the number of iterations needed for convergence of the proposed clustering method when applied on the image shown in Figure 4.20–i using the parameters of $c = 2$ and $\varepsilon = 0.5$ with varying values of m. Note that the elapsed time curve and the iteration count curve have the same shape. Thus, the convergence time of the method changes with the iteration count by just a factor, as expected. Also, an increasing m has resulted in

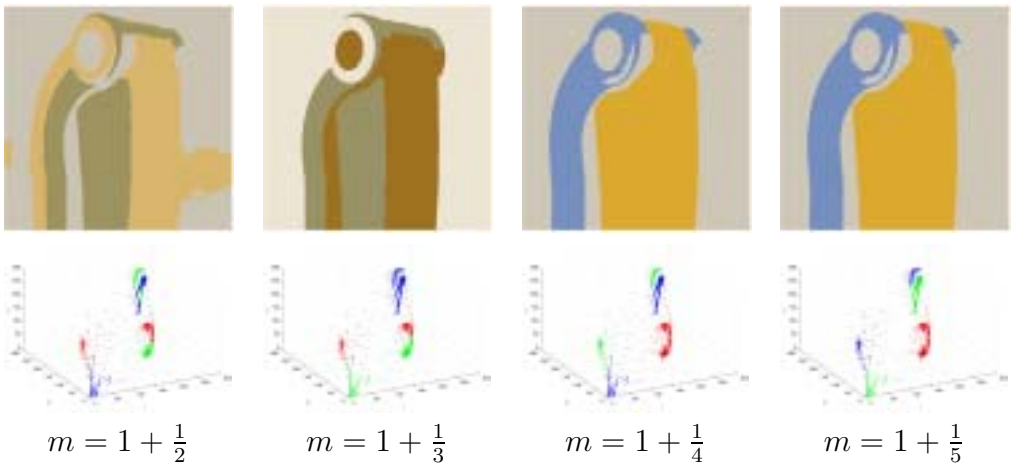
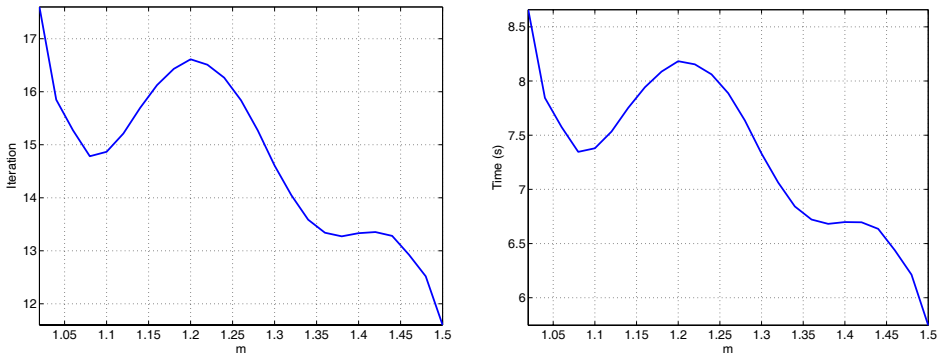


Figure 4.30: Results of different fuzziness values on the clusters corresponding to the image shown in Figure 4.20–f.

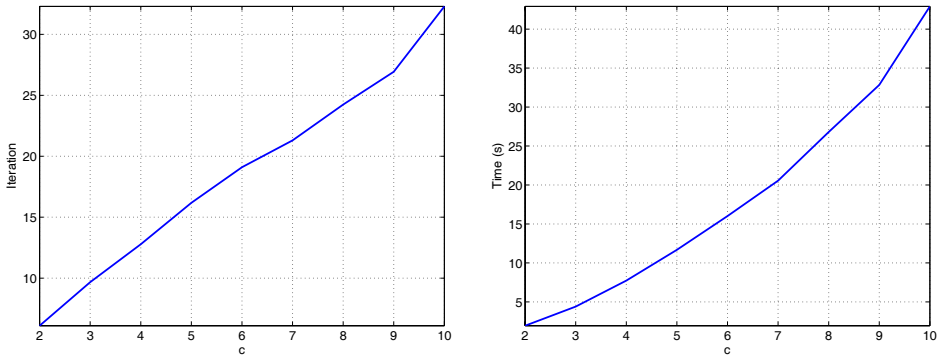
overall inflated number of iterations, while the whole operation is too fast (less than $9s$ in all tests), with m nonlinearly effecting it. Figure 4.32 shows the elapsed time and the iteration count for the same image, when clustered into different numbers of clusters. Note that the iteration count depends almost linearly on the number of clusters, while the elapsed time depends quadratically on it, (because of the increased computation cost of the membership values). Again, note the time record of less than a minute for the worth case of finding 10 clusters.

Figure 4.33 shows the results of the proposed segmentation method with the c parameter listed in Table 4.8. In this table, t_1 shows the time elapsed by the proposed segmentation method, while t_2 shows the time elapsed by the conventional FCM approach in the units of seconds. Figure 4.34 shows the corresponding computed



(a) (b)

Figure 4.31: Computational cost and iteration count of the proposed clustering method for different values of m for the image shown in Figure 4.20–i. (a) Iteration count, (b) Elapsed time.



(a) (b)

Figure 4.32: Computational cost and iteration count of the proposed clustering method for different values of c for the image shown in Figure 4.20–i. (a) Iteration count, (b) Elapsed time.

clusters in the RGB space. Here, the radius of the convolution kernel is selected to be 5. Investigating Table 4.8, shows the high time performance of the proposed method. We emphasize the low spatial computation of the proposed method. Figures 4.35 and 4.36 show the segmentation results and cluster shapes, produced by the conventional FCM, respectively. Comparing the results of the proposed segmentation method and the FCM approach, reveals the high performance of the proposed method. While for Figures 4.20–a, 4.20–b, and 4.20–c, the results of the two methods are almost the same, (both in the spatial domain and the spectral domain), considering the results on figure 4.20–d is important to understand the characteristics and differences of the two methods. The FCM has failed to distinguish between the two red and yellow cloths, (because of their close spectral zones). Also, in the Figure 4.20–e, the FCM has mistakely classified the apple and the cucumber in the same group. Also, in the case of Figure 4.20–g, the FCM has wrongly gathered some parts of the chimney and the red ribbon in the same group, and also the grass and some parts of the sand. In Figure 4.20–h, the FCM has failed to separate the points belonging to the wall and the kerchief, while in all these cases the proposed segmentation method has resulted perfectly. In Figures 4.20–i and 4.20–j, there are mistakes in the segmentation results of the FCM, (in distinguishing the face and the cloth and classifying the blue shades of the sky in the windows and the balcony ceils, respectively). Furthermore, in Figure 4.20–k the FCM has not been able to partition the flowers completely,

Table 4.8: Segmentation parameter and the computational cost of the proposed (t_1) and the FCM method [60] (t_2). [m: Fuzzyness, t_1: proposed method, t_2: the FCM method [60]]

Image	c	$t_1(s)$	$t_2(s)$
4.20–a	3	12.2	6.8
4.20–b	3	8.4	8.5
4.20–c	4	11.2	13.8
4.20–d	4	7.0	8.6
4.20–e	4	12.0	19.9
4.20–f	3	3.1	3.6
4.20–g	4	15.8	15.7
4.20–h	3	4.6	13.0
4.20–i	4	7.5	4.9
4.20–j	3	2.5	9.1
4.20–k	3	3.1	6.15
4.20–l	4	17.4	14.7

compared to the perfect results of the proposed method. Also, investigate the poor segmentation results of the FCM on the salad image in Figure 4.20–l, in which the carrot and the vegetables are put in the same class. Investigating the cluster shapes (Figures 4.34 and 4.36), reveals the reason of the FCM acting poor in color fields, (FCM tries to find spherical clusters, while color clusters are not spheres). Table 4.8 shows that the proposed method most of the times is also faster than the FCM, specially in the cases that FCM tries to cover cylindrical clusters with spheres and fails to comply with human perception. Note that in both methods, the values of the parameters are set as $m = 1.05$ and $\varepsilon = 0.5$.

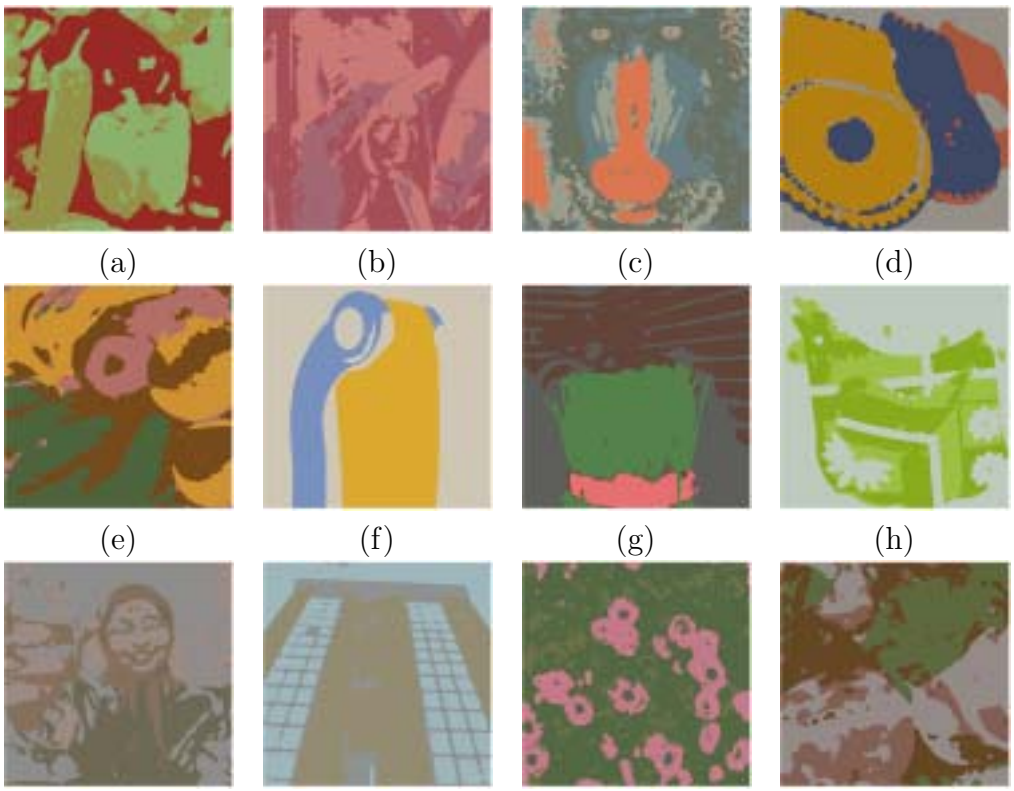


Figure 4.33: Results of applying the proposed segmentation method on the test images shown in Figure 4.20.

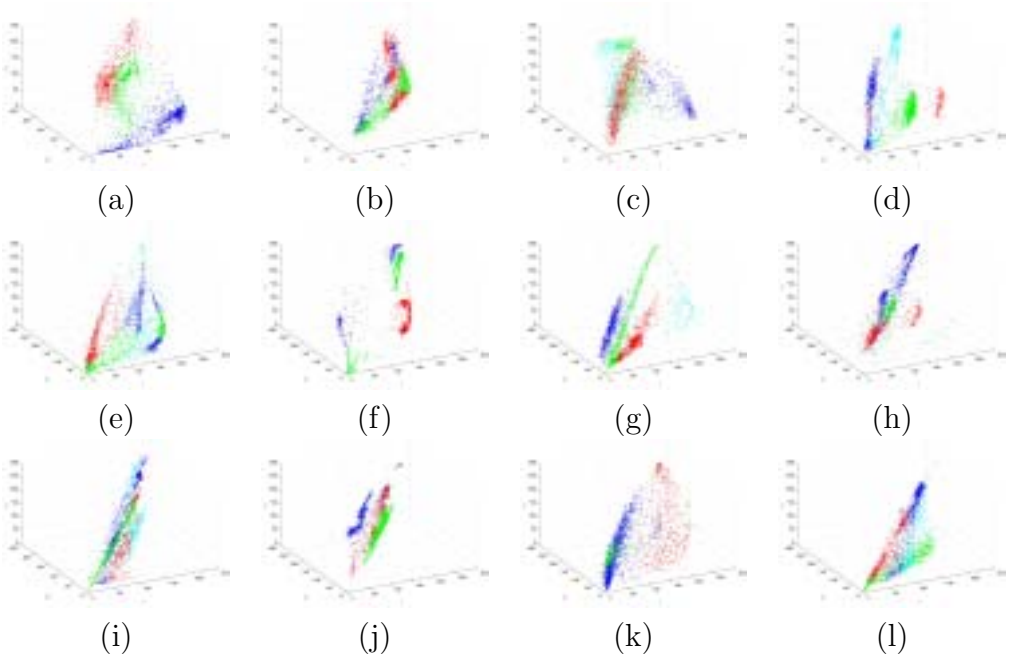


Figure 4.34: Shapes of the clusters corresponding to the segmentation results shown in figure 4.33, computed by the proposed method.

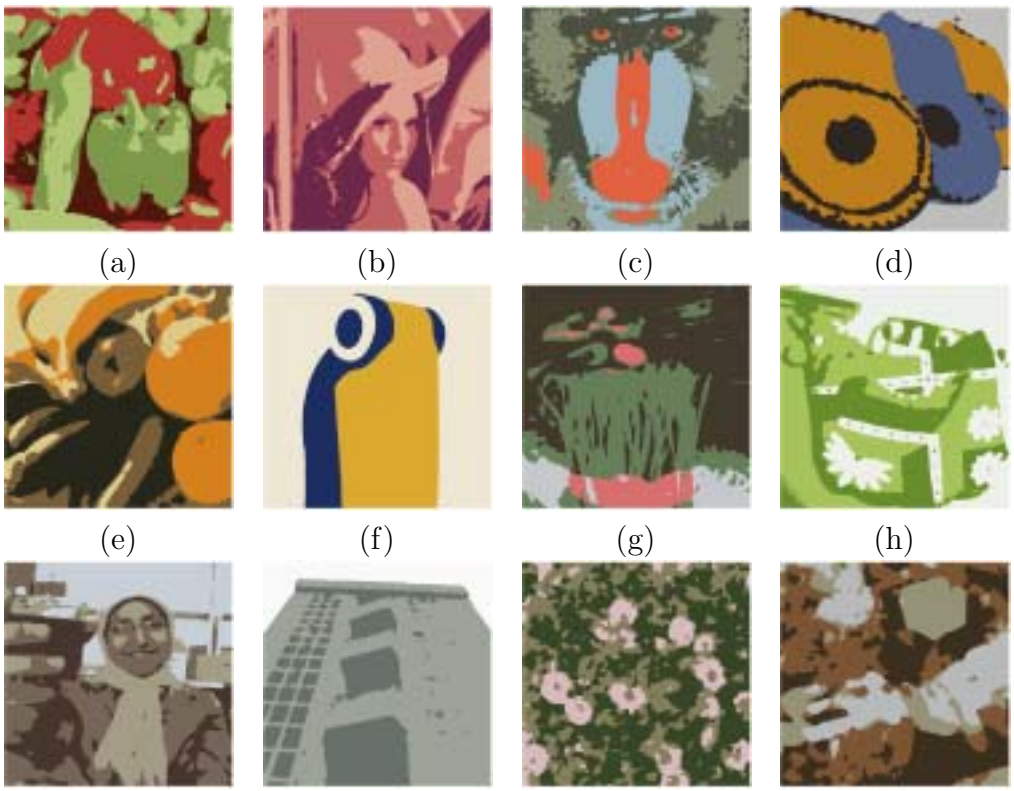


Figure 4.35: Results of applying the conventional FCM method on the test images shown in Figure 4.20.

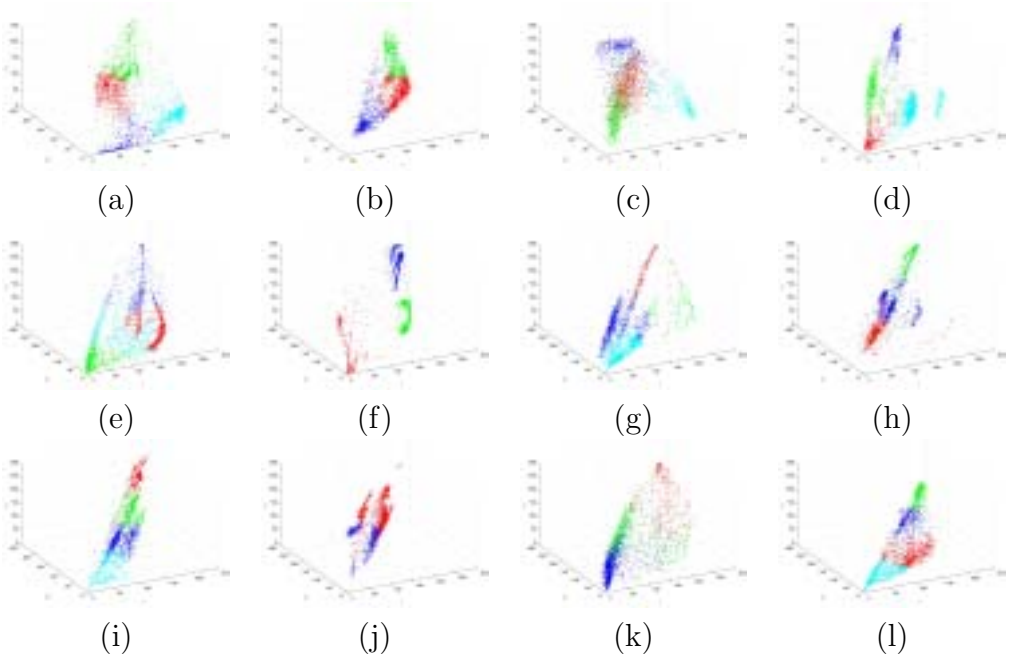


Figure 4.36: Shapes of the clusters corresponding to the segmentation results shown in figure 4.35, computed by the FCM method.

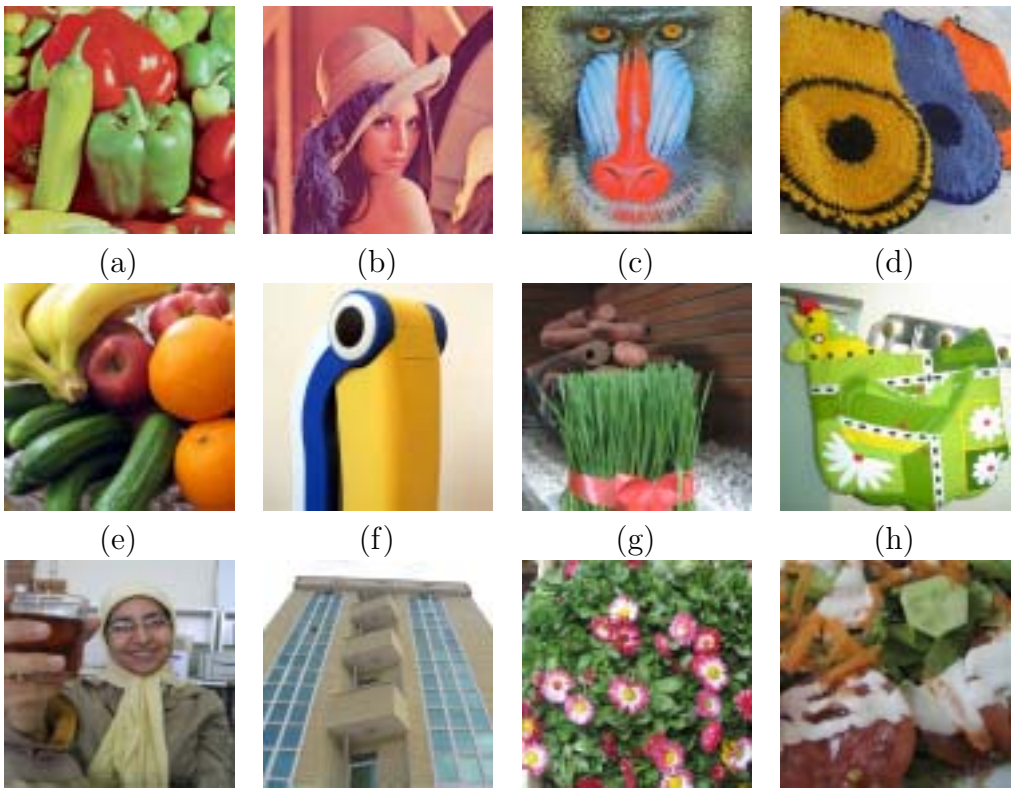


Figure 4.37: Some of the test images, (a)–c Standard images, (d)–l Adopted from [318].

4.6.2 Agent–based FPCAC

Figure 4.37 shows some of the test images. Figure 4.39 shows the values of J and dx in consecutive iterations and Figure 4.38 illustrates the corresponding clusters and segmentation results. Here, the clustering method is performed on the image shown in Figure 4.37–a and the parameters are set as $\varepsilon = 1$, $c = 2$, and $m = 1.05$. As shown in Figure 4.38, after the first iteration, the desired result is almost reached. Also, note the decline of J by factor of about 19 in the first iteration while at the fifth iteration the value of J does not change significantly. In this example, setting $\varepsilon = 5$, does not disturb the performance of the method, while it enhances the algorithm's speed (20%). Figure 4.40 and Figure 4.41 show the results for the image shown in Figure 4.37–g with the parameters set as $\varepsilon = 5$, $c = 3$, and $m = 1.05$. In this case J is almost divided by a factor of 12 in the first iteration, while it remains almost stationary in the fifth iteration. In the same manner, setting $\varepsilon = 5$ does not result in spurious results here. The same results are observed in other sample images. In the forthcoming parts of this thesis, the ε is always set equal to 5.

Figure 4.42 shows the results of the proposed method applied on the image shown in figure 4.37–d in three different races. Here, c is selected equal to 4. The detailed results are listed in Table 4.9. Note that although the values of m and J are different in different races, the final result and the integrity are the same. As seen in this test, and also observed in others, the value of i is more distinguishing that the value of

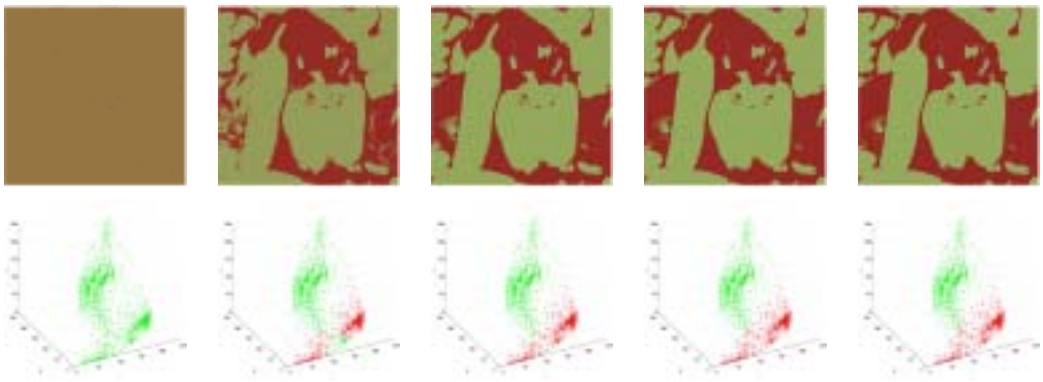
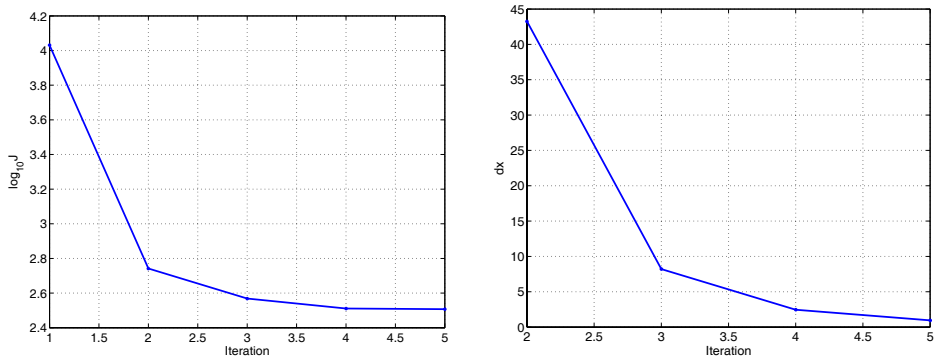


Figure 4.38: Consecutive iterations of one set of cooperating agents working on the image shown in Figure 4.37–a. Top: the segmentation results. Bottom: clusters.



(a) (b)

Figure 4.39: Results of the proposed clustering method for the image shown in Figure 4.37–a. (a) Decline of $\log_{10} J$ in consecutive iterations. (b) dx values corresponding to each iteration.

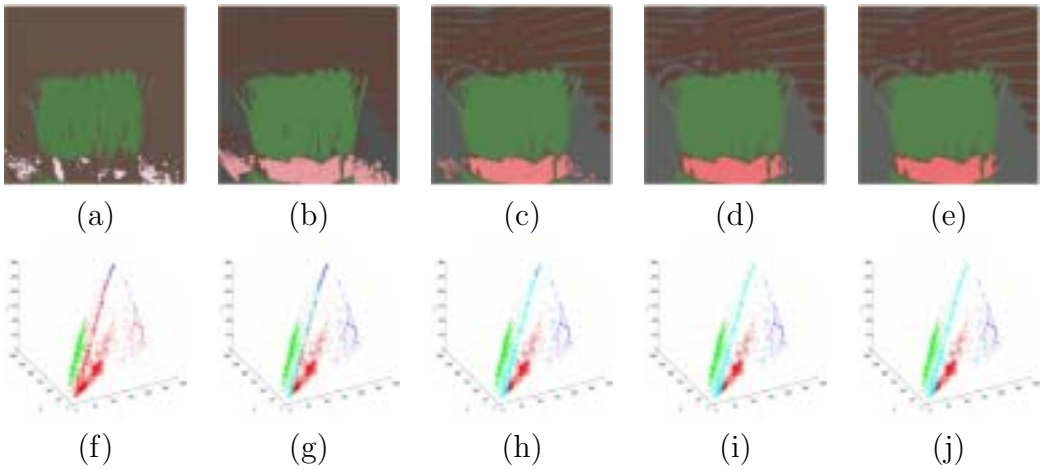


Figure 4.40: Consecutive iterations of one set of cooperating agents working on the image shown in Figure 4.37–g. Top: the segmentation results. Bottom: clusters.

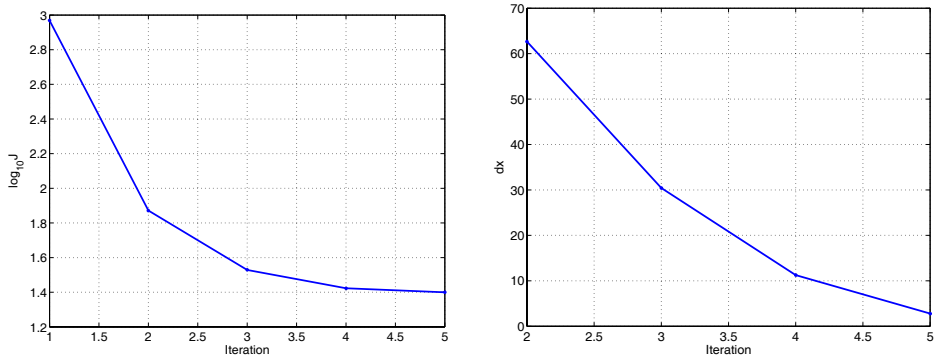


Figure 4.41: Consecutive iterations of one set of cooperating agents working on the image shown in Figure 4.37–g. Top: the segmentation results. Bottom: clusters.

Table 4.9: Results of three different races on the image shown in Figure 4.37–d. [t: Elapsed Time, J: Objective Function Value. i: Integrity.]

m	$t(s)$	J	i
1.14	200	41.7	1.25
1.61	220	32.5	1.25
1.81	200	31.0	1.25

Table 4.10: Results of the proposed clustering method performed on the image show in Figure 4.37–e with different values of c. [t: Elapsed Time, J: Objective Function Value. i: Integrity. c: Cluster Count.]

m	$t(s)$	J	i	c
1.86	76	521	1	2
1.21	140	262	1	3
1.81	170	97	1.25	4
1.32	250	64	1.2	5
1.65	340	56	1.17	6

J. This test along with other observations on other sample images shows the high repeatability of the proposed method.

Figure 4.43 shows the results of the proposed clustering method performed on the image shown in Figure 4.37–e with different values of c. Detailed results are listed in Table 4.10. Note that altering c controls the resolution of the clustering, while in all cases the result is perceptually valid. Also, note that the elapsed time of the method depends quadratically on c and that the value of J drops when c increases, and thus its is not comparable for sets of cooperative agents with different sizes. The same happens for the value of i.

Figure 4.44 shows the results of the proposed clustering method applied on the

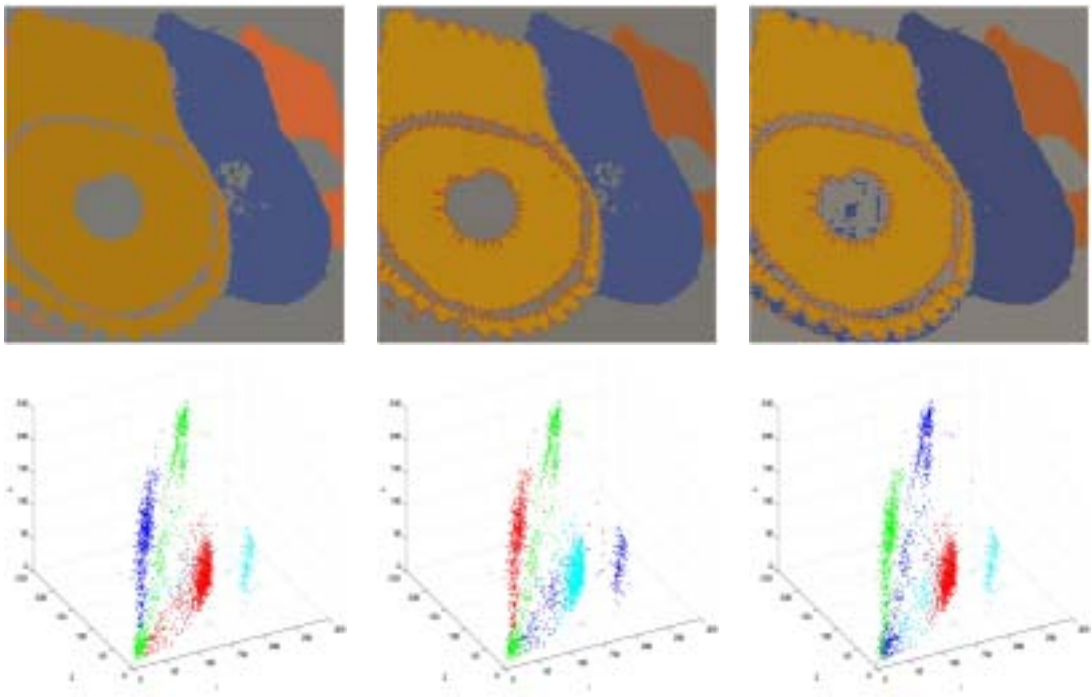


Figure 4.42: Results of three different races on the image shown in Figure 4.37–d.

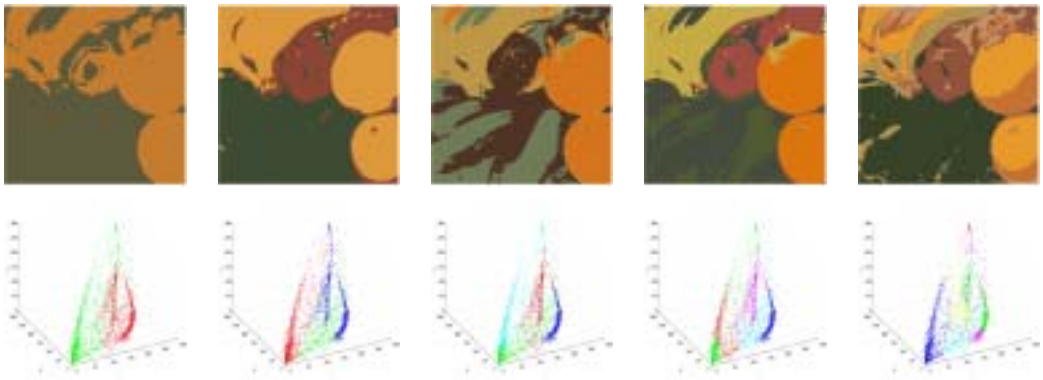


Figure 4.43: Results of the proposed clustering method performed on the image show in Figure 4.37–e with different values of c.

images shown in Figure 4.37 using 50 agents sets. The detailed results are shown in Table 4.11. note that for c clusters, the value of $i \leq 1 + \frac{1}{c}$ means that, at least $c-1$ agents have reached to single clusters, while at most one has experienced a split cluster. This event have a meaningful description; all of the agents except only one have found the real clusters, and the remainder's territory includes the residual points (the background). Investigating Table 4.11 shows that this is the case for all samples, except for the image shown in Figure 4.37–i, in which two agents out of seven have failed, while the final clustering results are satisfactory. Also, note that the value of J for different images, even with the same value of c, are not comparable. This means that no general threshold on J can be defined. Also, note that the range of the values of m almost fills the entire $[1^+, 2]$ interval, meaning that the initial selection of the range for m can not be limited.

4.6.3 Tree Decomposition

To compare the performance of the quad–tree and the bi–tree decomposition methods, each sample images is decomposed by each method using values of $\varepsilon_1 = 1, \cdots, 15$ and $\varrho = 1, \cdots, 5$, and the results are recorded. Figure 4.45 shows values of $\tilde{\varrho}$ for different values of ϱ and ε_1 in the two methods (left: quad–tree, right:bi–tree) for the *Peppers* standard image. Figure 4.45, shows that in both algorithms in all sets of parameters we have $\tilde{\varrho} \geq \varrho$, with the equality in smaller values of ε_1. Also, as desired, $\tilde{\varrho}$ is a monotonic non–increasing function of ε_1. Inspecting the falling tender of the

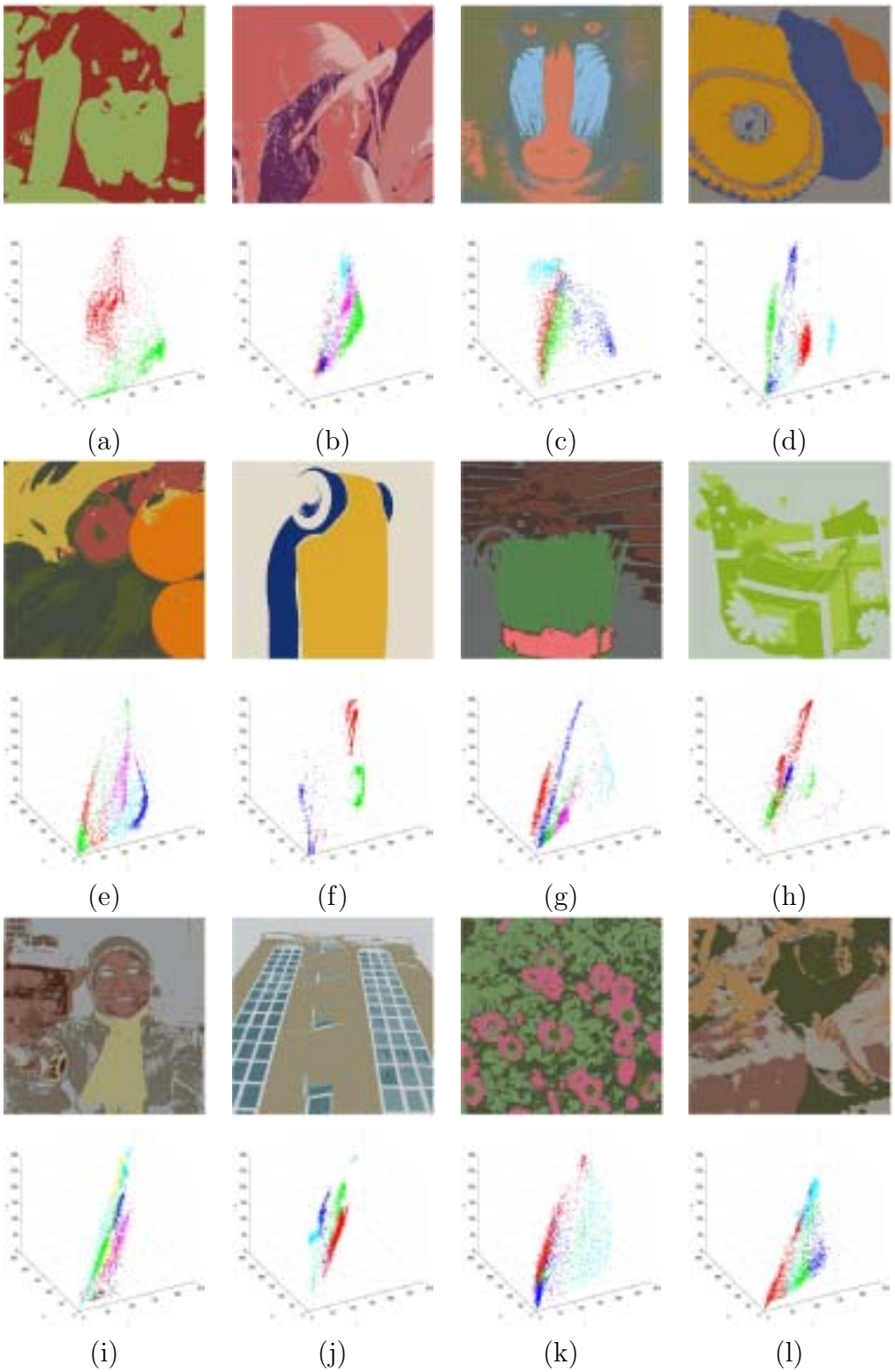


Figure 4.44: Results of the proposed clustering method performed on the images shown in Figure 4.37.

Table 4.11: Results of the proposed clustering method performed on the images shown in Figure 4.37. [m: Fuzziness. t: Elapsed Time, J: Objective Function Value. i: Integrity.]

Image	m	$t(s)$	J	i	c
4.37–a	1.71	28	308	1	2
4.37–b	1.1	200	17	1	5
4.37–c	1.09	160	110	1.25	4
4.37–d	1.81	200	31	1.25	4
4.37–e	1.32	250	64	1.2	5
4.37–f	1.26	140	59	1.33	3
4.37–g	1.34	250	16	1	5
4.37–h	1.81	140	96	1.33	3
4.37–i	1.3	300	7	1.29	7
4.37–j	1.35	150	16	1.25	4
4.37–k	1.83	180	56	1.25	4
4.37–l	1.79	200	29	1	4

$\tilde{\varrho}$ with increasing ε_1 gives the promising result, that although in both methods, by increasing the marginal acceptable non–homogeneity (relieving the method), the average splitting decreases, bi–tree gives less splitting in average. For example, compare the number of blocks at $\varepsilon_1 = 10$ for each set of corresponding curves in the left and the right images. For the quad–tree the values are approximately {1.00, 2.00, 2.80, 3.61, 4.23}, while the corresponding results for the bi–tree method are {1.00, 1.97, 2.76, 3.35, 3.77}, that are about 9% better in the largest ρ.

Figure 4.46 shows the values of $\log_4 n$ for different values of ϱ and ε_1 in quad–tree and bi–tree methods for the *Peppers* image. While, as expected, n is a monotonic non–increasing function of ε_1, there are less blocks produced by bi–tree compared with the quad–tree, when used in the same conditions. Namely, in the case of ε_1,

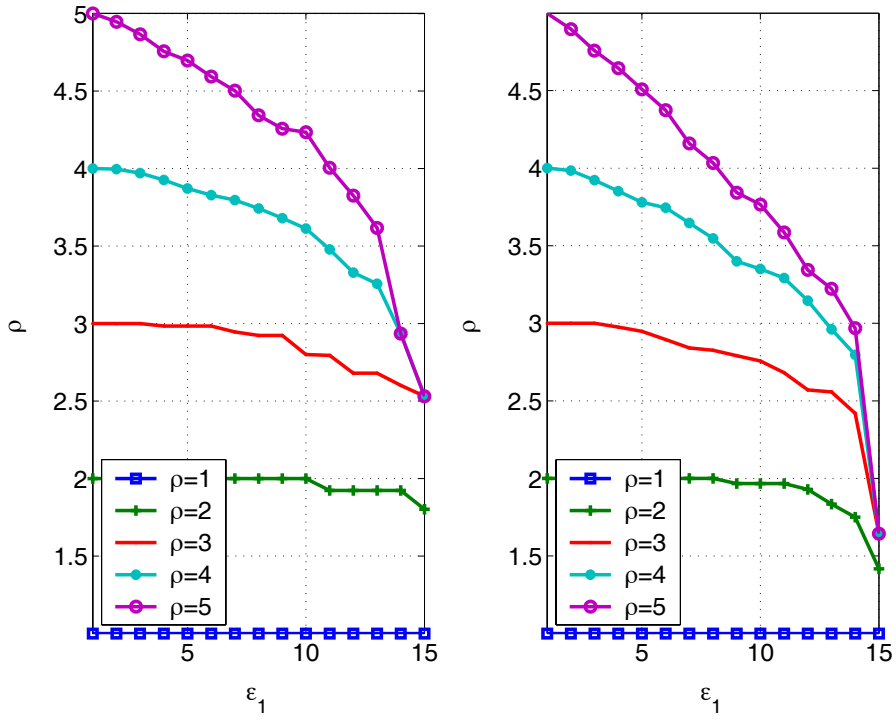


Figure 4.45: Values of $\tilde{\varrho}$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree method and right: proposed bi–tree method.

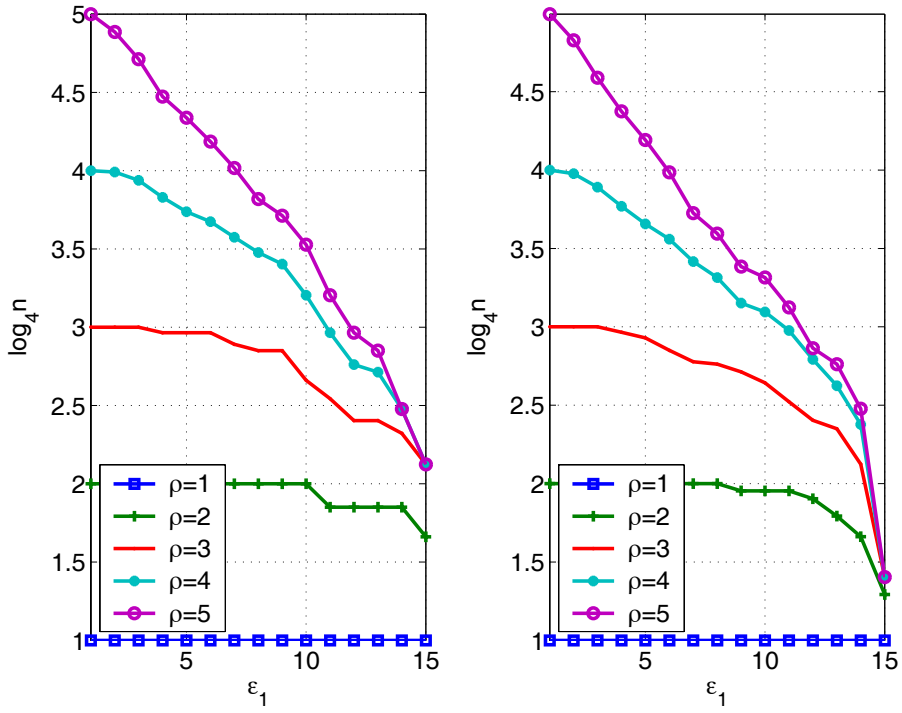


Figure 4.46: Values of $\log_4 n$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree.

bi–tree has produced $\{4, 15, 39, 73, 99\}$ blocks for consecutive values of ρ, while quad–tree has produced $\{4, 16, 40, 85, 133\}$ blocks. Note that when tree depth increases, quad–tree results in much more block numbers (34% more for $\rho = 5$). This inflated number of blocks has a negative impact on the speed performance of the afterwards stages of the process.

Figure 4.47 shows the values of $\tilde{\varphi}$ for different values of ϱ and ε_1 in the two methods for the *Peppers* image. As shown in Figure 4.47, in both methods, $\tilde{\varphi}$ is always less than 1.3. While the range of $\tilde{\varphi}$ in bi–tree occupies the $[1, 1.2]$ interval, quad–tree

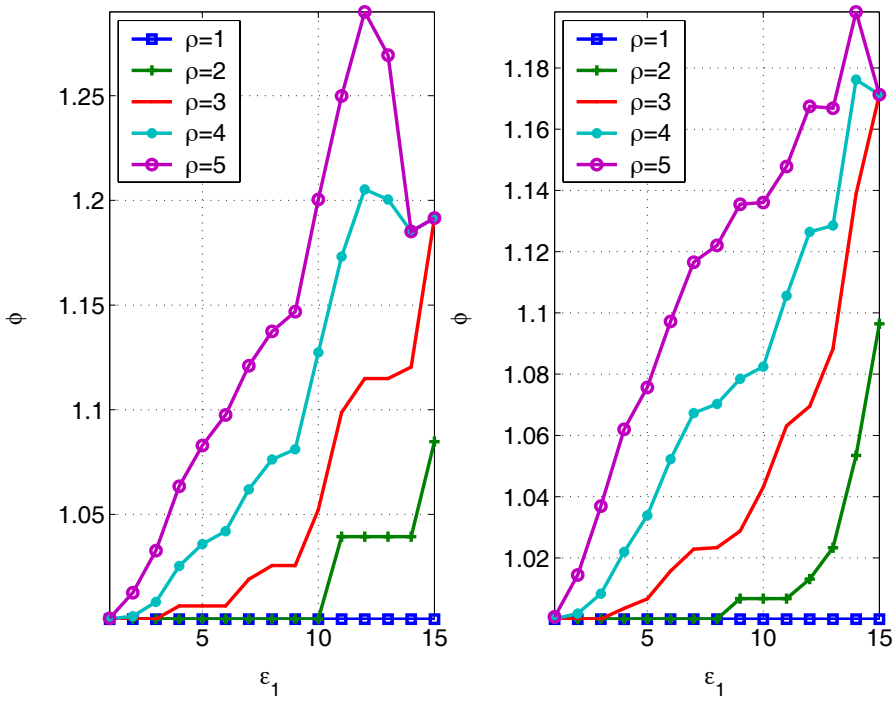


Figure 4.47: Values of $\tilde{\varphi}$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree.

produces $\tilde{\varphi}$ values in the range of $[1, 1.3]$. Thus, the size scatter is about 50% wider in quad–tree when compared to the bi–tree. This means that the result of the quad–tree is more non–homogenous in its size distribution compared to the bi–tree. When using the proposed block–wise interpolation, this scattered pattern of block sizes results in visual artifacts in the images.

Figure 4.48 shows the values of $\frac{\tilde{\varepsilon}_1}{\varepsilon_1}$ for different values of ϱ and ε_1 in the *Peppers* standard image using the two decomposition methods of quad–tree and bi-tree. The mean square difference between corresponding points of these two sets of curves is

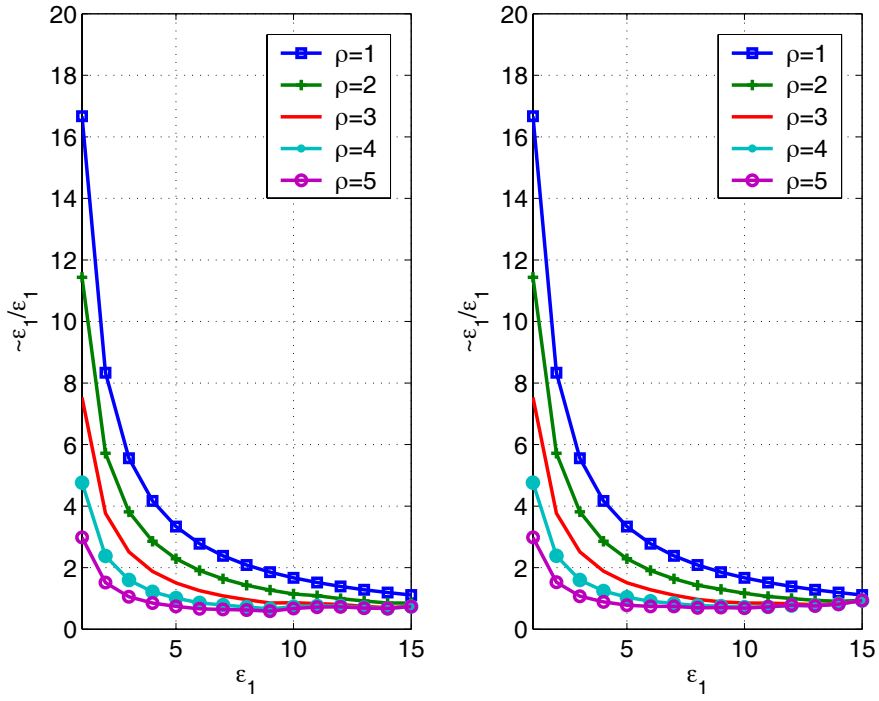


Figure 4.48: Values $\frac{\tilde{\varepsilon}_1}{\varepsilon_1}$ for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree.

0.059. Thus, in terms of final homogeneity, the two tree decomposition methods are almost identical.

Figure 4.49 shows the time elapsed by the two decomposition methods processing the *Peppers* standard image with different values of ϱ and ε_1. Note that the proposed bi–tree elapses almost three times more computation cost. This happens due to the decision stage in the proposed bi–tree.

Thus, comparing bi–tree and quad–tree decomposition methods, bi–tree produces less blocks with almost identical sizes, while quad-tree splits the image more with a

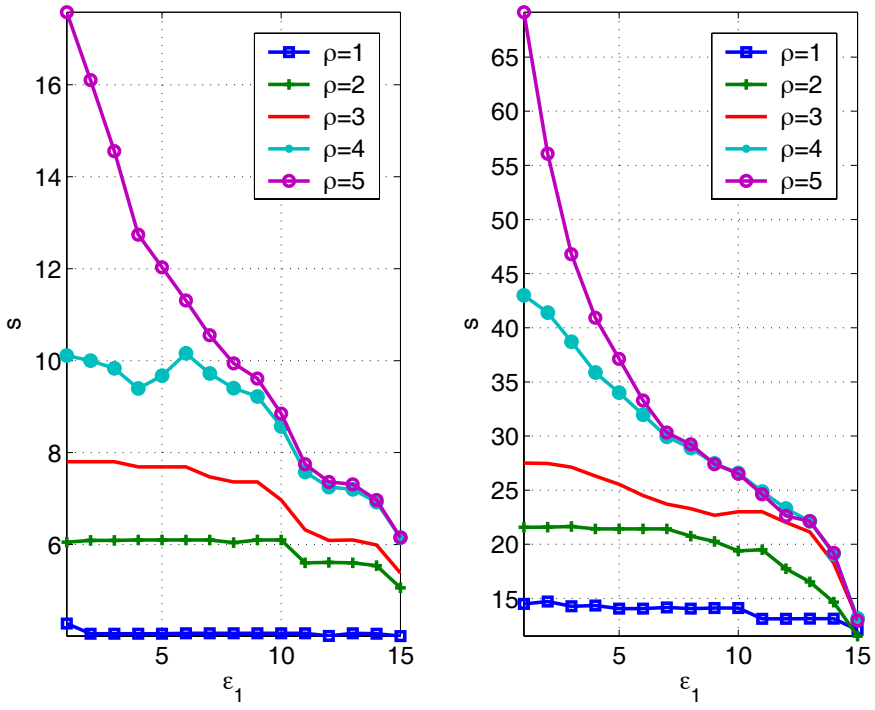
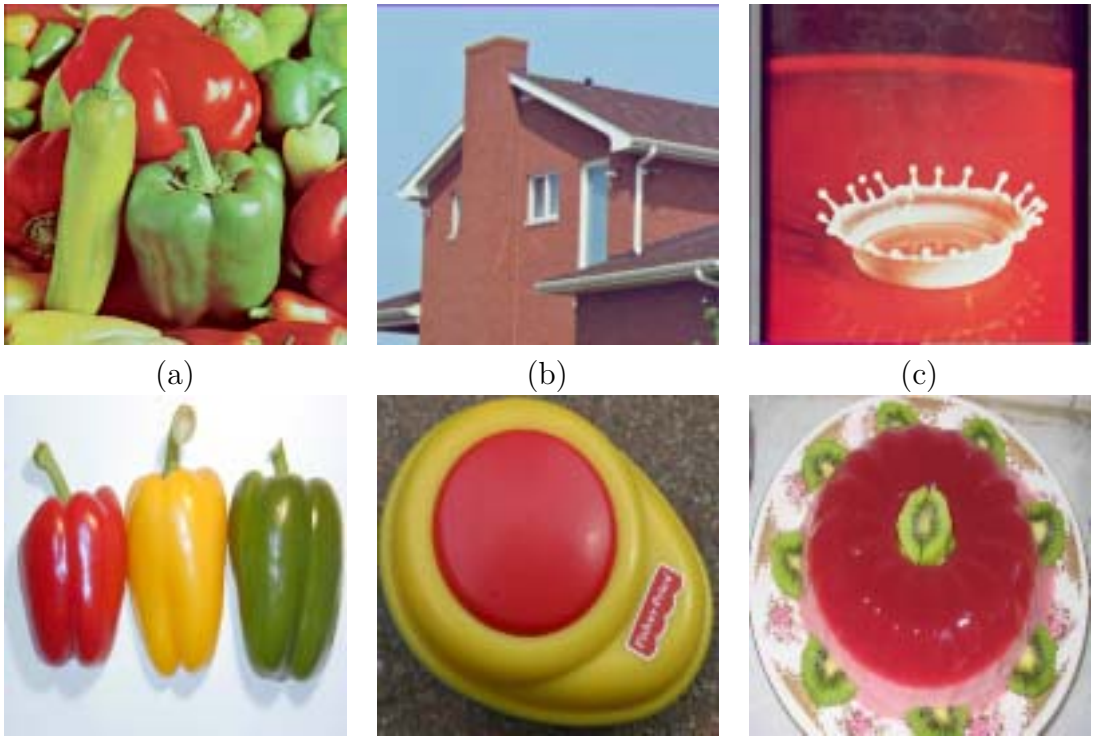


Figure 4.49: Elapsed time of the decomposition methods for different values of ϱ and ε_1 in the *Peppers* image, left: quad–tree and right: bi–tree.



(a) (b) (c)

(d) (e) (f)

Figure 4.50: Sample images. a) *Peppers*. b) *House*. c) *Splash*. a,b,c) Courtesy of USC–SIPI, Signal and Image Processing Institute at the University of Southern California. d) *Food Peppers*, courtesy of *freeimages.co.uk*. e) *Toy*, courtesy of Computational Vision at Caltech, *http://www.vision.caltech.edu*.

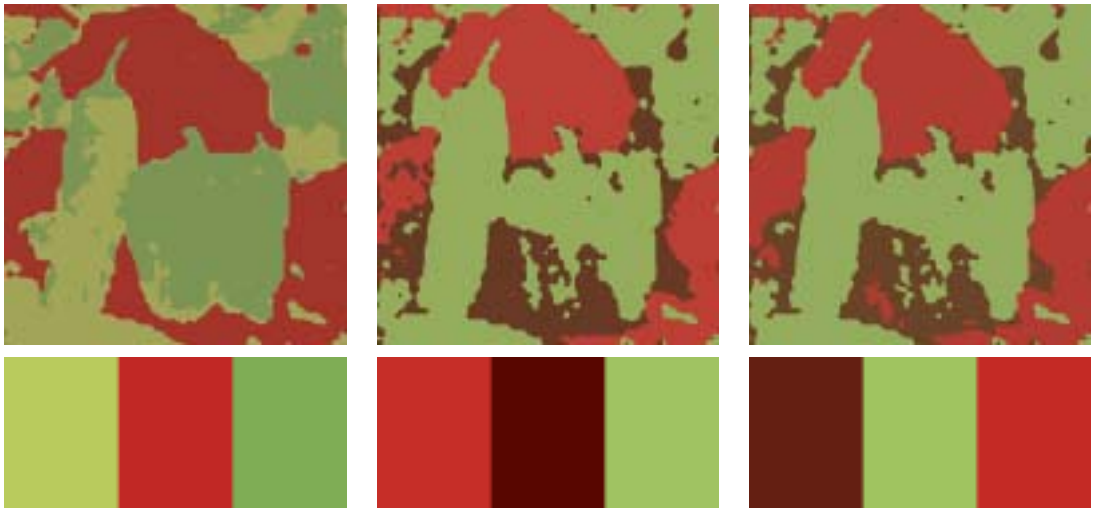
more scattered pattern of block sizes. The total homogeneity of the resulting blocks are almost the same in both methods, while bi–tree is more time consuming. Further optimizations must be performed on the bi–tree implementation to increase its time performance.

4.6.4 Principal Colors

Based on the previous research, the value of θ is always set equal to 10.

Figure 4.51 shows the results of three times applying the proposed method on the image *Peppers*. Here, $m = 2$, $c = 2$, $n_1 = 4$ and $n_2 = 3$ are used. This figure shows that while the proposed method always results satisfactorily, like the entire fuzzy clustering category, its output is never identical in different runs, even with the identical values of parameters. It should be emphasized that in contrast with the conventional pixel–based fuzzy clustering approaches, in which local minima is an unsolved problem [61], the proposed method never falls into a local minimum. We believe that this is the result of two innovations in the proposed method. First, rather than facing the original realizations, which are noisy and contain many outliers, the proposed method works on more robust statistical parameters of blocks of the image. Secondly, the proposed distance function completely fits the physical phenomenon which is analyzed. The non–entrapment of the proposed method in local minima gives it a desirable habit; the proposed method acts more repeatable compared with the FPCAC which uses a similar distance measure on singular pixels. In these experiments the algorithm has converged in 9, 6, and 5 iterations , elapsing $16s$, $14s$, and $14s$, respectively.

In another experiment we investigate the effect of m. Figure 4.52 shows the results of the proposed method on the image *House*. Here, $c = 3$ is used with $m = 1\frac{1}{3}$, $m = 1\frac{2}{3}$, and $m = 2$. The algorithm has converged in 4, 10, and 9 iterations, elapsing $13s$, $16s$, and $16s$, respectively. More experiments show that the actual value of m



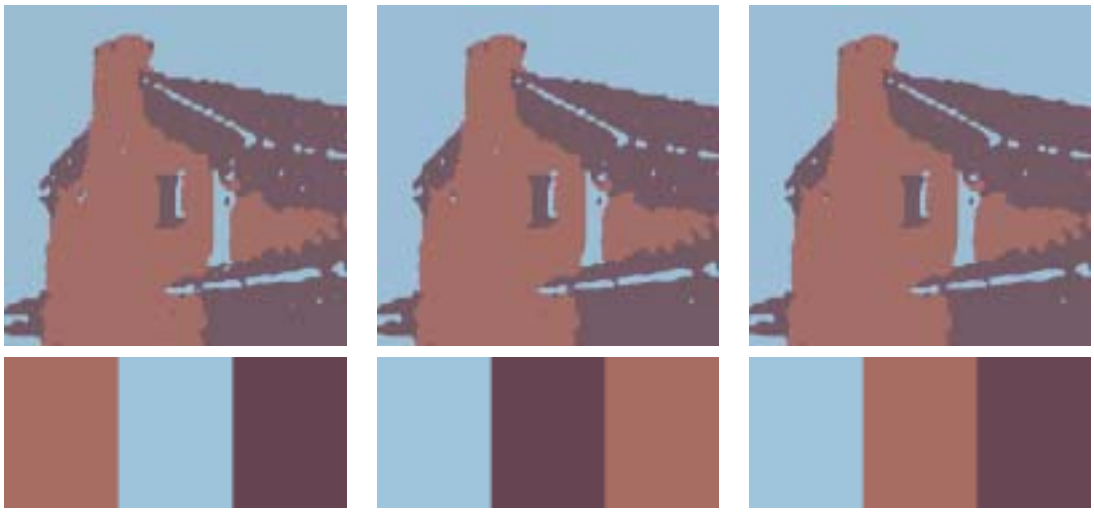
(a) (b) (c)

Figure 4.51: Results of three times application the proposed method on the image *Peppers*. Top) Results of segmentation. Bottom) Principal colors.

has no visible effect on the results of the proposed method. Hence, the value of $m = 2$ is used as the default value. Note that the values of n_1 and n_2 only affect the spatial precision of the results and the elapsed time. As a compromise, we select $n_1 = 4$ and $n_2 = 3$, while many other values are easily applicable.

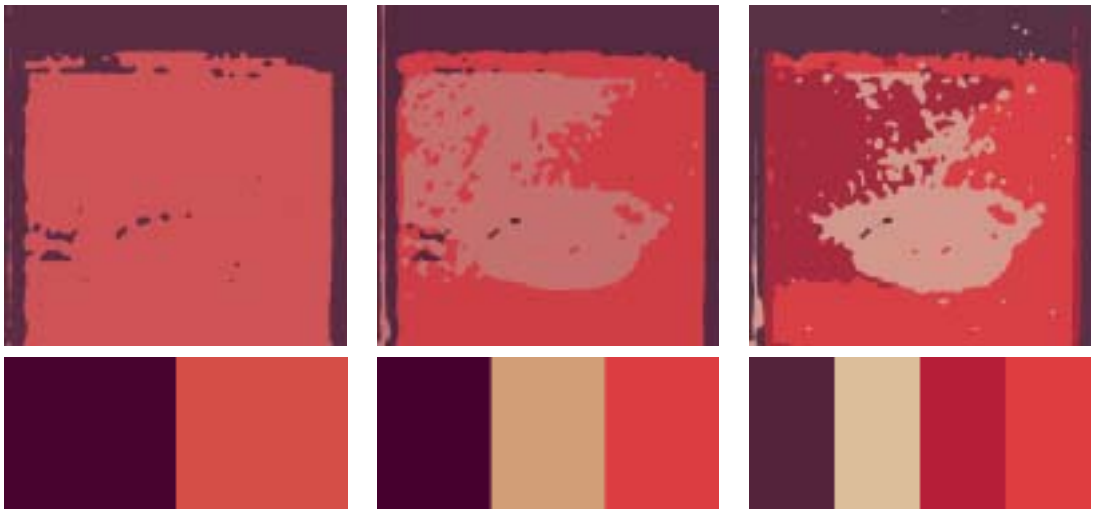
Figure 4.53 shows the effect of choosing different values of c. As expected from a proper segmentation algorithm, increasing the number of clusters results in a more detailed segmentation. The algorithm has converged in 9, 5, and 8 iterations , elapsing $12s$, $14s$, and $18s$, respectively. It should be emphasized that the elapsed time depends almost linearly to the number of clusters.

Figure 4.54 compares the proposed method with the FPCAC (see Section 3.5.2).



(a) (b) (c)

Figure 4.52: Effects of m on the results of the proposed method on the image *House*. Top) Results of segmentation. Bottom) Principal colors.



(a) (b) (c)

Figure 4.53: Effects of c on the results of the proposed method on the image *Splash*. Top) Results of segmentation. Bottom) Principal colors.

Table 4.12: Comparison of the proposed method with the FPCAC described in Section 3.5.2. [t: Elapsed time. i: Number of iterations before convergence.

Image	**Proposed Method**		**FPCAC**	
	$t(s)$	i	$t(s)$	i
Figure 4.50–d	20	10	26	40
Figure 4.50–e	18	6	8	19
Figure 4.50–f	21	11	24s	38

This comparison is performed because both of the methods search for cylindrical structures in the color space. While FPCAC works on individual pixels, the proposed method utilizes more robust statistically computed features. Comparison shows that the proposed method tends to give better results in an averagely smaller time. In this experiment, FPCAC is utilized with its own default value of $m = 1.05$. Table 4.12 shows the elapsed time and number of iterations of each method when working on each image. Also, the value of $c = 4$ is used for both method in all cases.

4.7 Compression

4.7.1 Up–sampling

To compare the results of the proposed compression method, 6 standard images of *Peppers*, *Couple*, *Girl*, *Lena*, *Airplane*, and *Mandrill* are compressed by setting $\varepsilon_1 = 1 \cdots 10$ for each of the three proposed decomposition methods (30 compressions per image). In all tests ϱ is computed. Figure 4.55 shows nominal results of the standard images. The PSNR and λ values are denoted in the caption. For different values of $\varepsilon_1 \in [0 \cdots 10]$, Figure 4.56-a shows values of the $PSNR_0$ and the $PSNR_+$ for

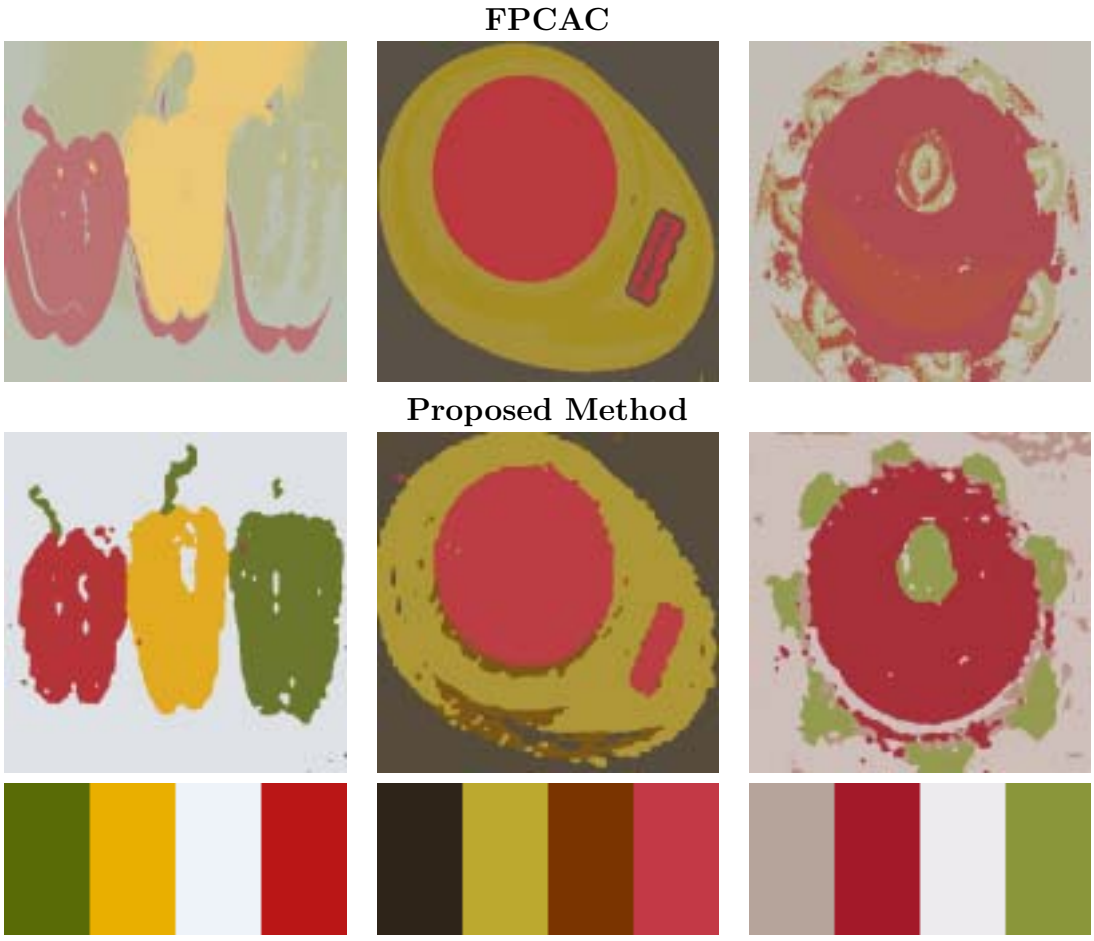


Figure 4.54: Comparison of the proposed method with the FPCAC described in Section 3.5.2.

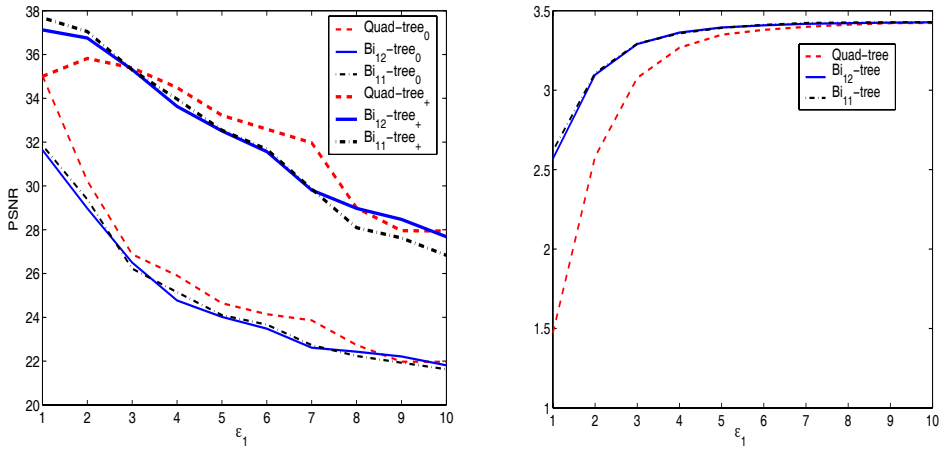
(a) (b) (c)

(d) (e) (f)

Figure 4.55: Results of the proposed compression method. (a) *Lena*: PSNR=35dB, $\lambda = 3.13$ (b) *Mandrill*:PSNR=31dB, $\lambda = 1.79$ (c) *Girl*: PSNR=35dB, $\lambda = 2.86$ (d) *Airplane*:PSNR=35dB, $\lambda = 3.13$ (e) *Peppers*:PSNR=31dB, $\lambda = 2.50$ (f) *Couple*:PSNR=36dB, $\lambda = 3.13$

the standard image *Lena* and Figure 4.56-b shows the respective compression ratio results.

The sample images of Figure 4.55 are subjectively desiring. Authors have performed a very simple but efficient test. Among the 31 people participating in a subjective test, no one is able to distinguish between the compressed images and the original ones. Even the author was many times in doubt, wether they are watching



(a) (b)

Figure 4.56: Results of the proposed compression method on standard image *Lena* for different values of ε_1. (a)$PSNR_0$ and $PSNR_+$, (b) Compression ratio

the original image or the compressed one.

λ values must be considered precisely. As the proposed compression method works on spectral redundancy, the marginal value of λ is 3. Having in mind that almost no spatial redundancy decreasing have performed in this method, any gray-scale image compression method can be serialized with the proposed method, to compress the 7-bit, G matrix. The point is the definitely independence of the spatial and spectral compression methods.

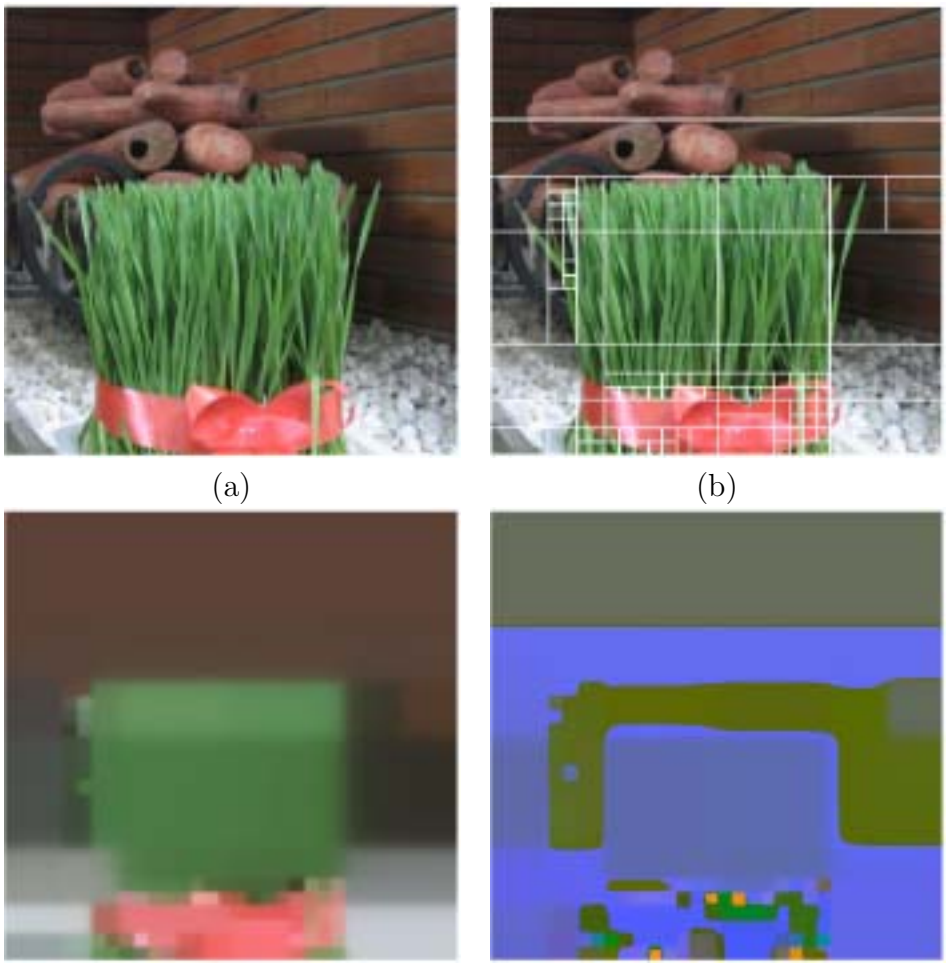
As shown Figure 4.56-a, bi_{11}-tree over-performs the bi_{12}-tree in terms of PSNR. Although, quad-tree has given better PSNR values compared with bi-tree methods for higher values of ε_1, considering that the high compression ratios are gained in lower values of ε_1, where bi_{11}-tree is the dominant method, proves that bi_{11}-tree is

the best selection for compression. A noticeable fact in Figure 4.56-a is the same value of the PSNR for quad-tree in $\varepsilon_1 = 1$. As with decreasing values of ε_1, quad-tree decomposition process converges to the ordinary uniform sampling, the proposed compression method reduces to a simple sub-sampling of color information. the more than 2 steps difference between values of the PSNR of quad-tree and bi_{11}-tree at $\varepsilon_1 = 1$ shows the over-performance of the proposed bi_{11}-tree decomposition. The higher than 3 values of compression ratio with the PSNR value of more than $35dB$ qualifies the performance of the proposed compression method.

4.7.2 Block–Wise Interpolation

Assume the image shown in Figure 4.57–a, which is decomposed with parameters of $p = 0.5$, $\varepsilon_1 = 5$, and $\varrho = 5$ into 91 blocks (see Figure 4.57–b). Figures 4.57–c and 4.57–d show the corresponding EMap and RMap. Figure 4.58 shows another sample produced with the same values of the parameters (60 blocks). Figures 4.59 and 4.60 show the three pc_i channels corresponding to the images in figures 4.57–a and 4.58–a, respectively. In all eigenimages the dynamic range of the image is exaggerated to give a better visualization using the transformation in (3.7.189).

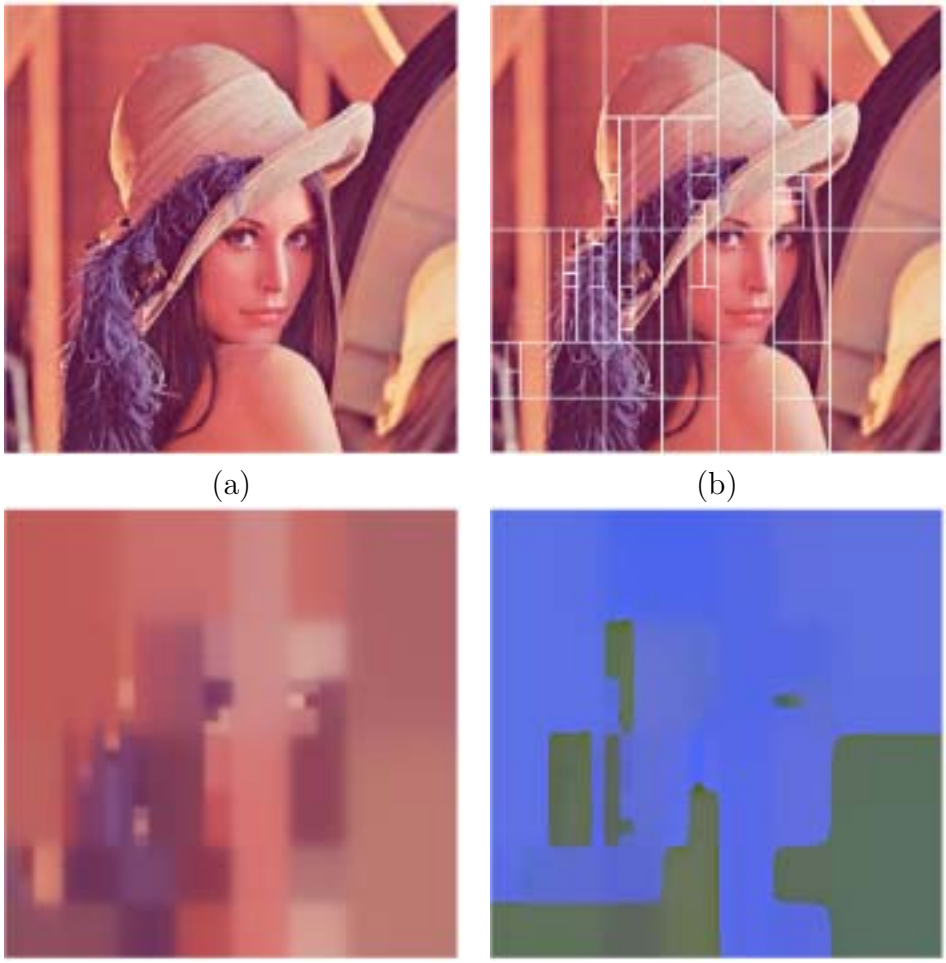
The stochastic distribution of pc_i is investigated in Figure 4.61 which shows the histogram of the three produced planes for the case of the image shown in Figure 4.57–a. In this example the standard deviations of the pc planes is computed as: $\sigma_{pc_1} = 52$, $\sigma_{pc_2} = 12$, and $\sigma_{pc_3} = 6$. Figure 4.62 shows the same for the image shown in



(a) (b)

(c) (d)

Figure 4.57: (a) Original image adopted from [318], (b) result of the bi–tree decomposition method, (c) Emap, and (d) Rmap.



(a) (b)

(c) (d)

Figure 4.58: (a) Original image , (b) result of the bi–tree decomposition method, (c) Emap, and (d) Rmap.

(a) (b) (c)

Figure 4.59: Eigenimages of the image shown in Figure 4.57–a, (a)pc_1, (b)pc_2, (c)pc_3.

(a) (b) (c)

Figure 4.60: Eigenimages of the image in Figure 4.58–a, (a)pc_1, (b)pc_2, and (c)pc_3.

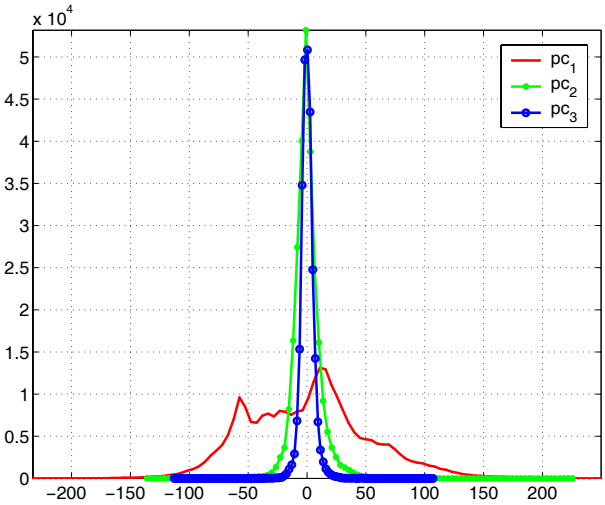


Figure 4.61: Histograms of pc_1, pc_2, and pc_3 for the image shown in Figure 4.57–a.

Figure 4.58, with standard deviation values of $\sigma_{pc_1} = 64$, $\sigma_{pc_2} = 12$, and $\sigma_{pc_3} = 4$. Note the perfect compaction of the energy in pc_1.

Figure 4.63–a, 4.63–b, and 4.63–c show the values of κ_1, κ_2, and κ_3 for the image in Figure 4.57–a for different values of ε_1 and ϱ. Note that rather than the trivial cases of $\varrho \leq 2$ and $\varepsilon_1 > 9$ (which are never used actually), more than 90% of the image energy is stored in pc_1, while pc_2 and pc_3 hold about 9% and 1% of the energy, respectively. Having in mind that in the original image $\kappa_r = 38\%$, $\kappa_g = 32\%$, and $\kappa_b = 30\%$, the energy compaction of the eigenimages are considerable.

Figure 4.64 shows the results of reconstructing the image of Figure 4.64–a from the eigenimages. While Figure 4.64–b shows the result of reconstructing the image using all three eigenimages, Figures 4.64–c and 4.64–d show the results of ignoring pc_3 and

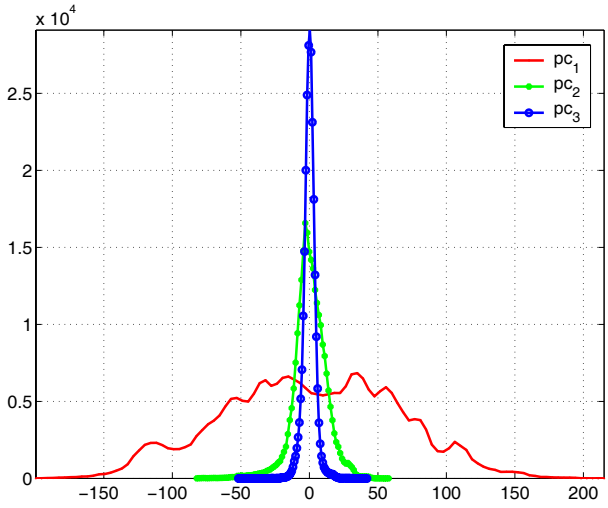
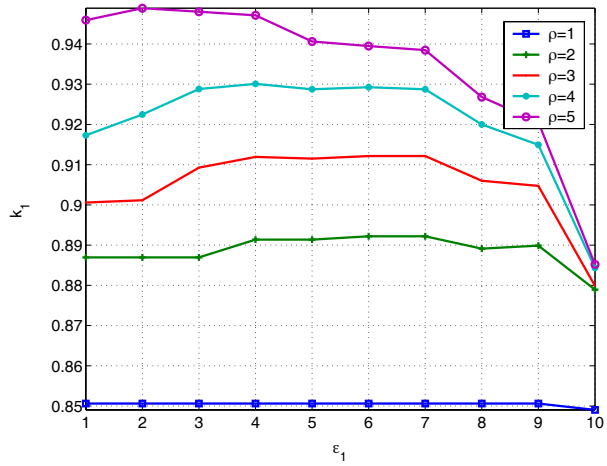


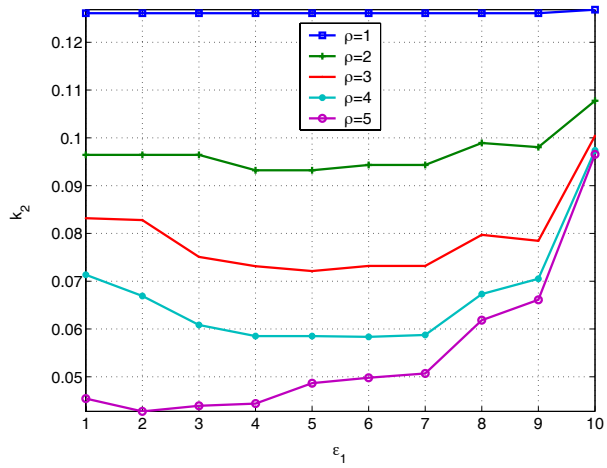
Figure 4.62: Histograms of pc_1, pc_2, and pc_3 for the image shown in Figure 4.58–a.

both pc_3 and pc_2, respectively. The resulting PSNR values are $60dB$, $38dB$, $31dB$. Note that PSNR of $60dB$ (instead of infinity), for reconstructing the image using all eigenimages is caused by the numerical errors, while the two other PSNR values ($38dB$, $31dB$) show some loss of information. Figure 4.65 shows the same results for the *Lena* image. The resulting PSNR values are $62dB$, $40dB$, and $31dB$, respectively. As indicated recently PSNR values of above $38dB$ are visually satisfactory even for professionals [207].

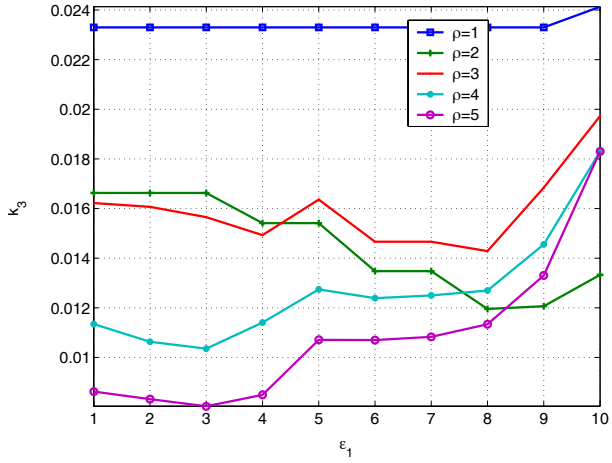
Figure 4.66 shows the PSNR values obtained by reconstructing the image using all the three channels (Figure 4.66–a), only two channels (Figure 4.66–b), and just one channel (Figure 4.66–c), for different values of ε_1 and ϱ. Note that for values of $\varepsilon_1 \leq 8$ and $\varrho \geq 3$, reconstructing the image using all eigenimages gives the high PSNR value



(a)

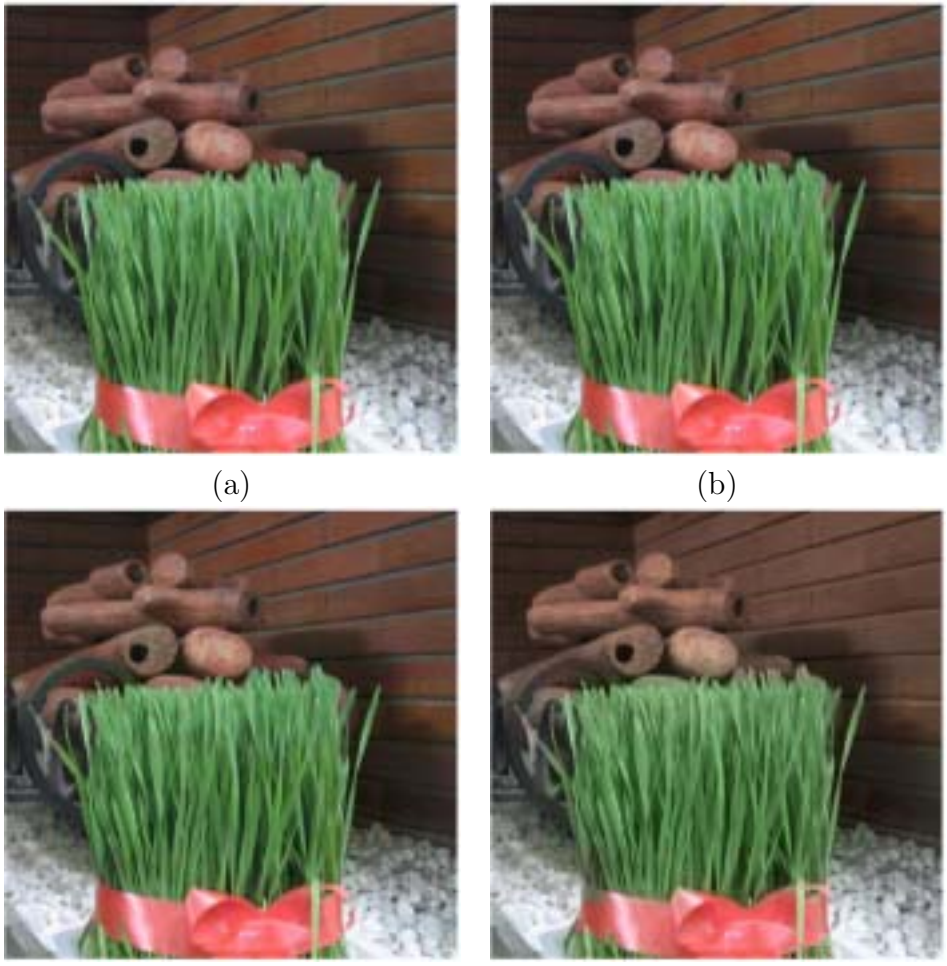


(b)



(c)

Figure 4.63: Distribution of the energy between the three eigenimages for different values of ε_1 and ϱ. (a)κ_{pc_1}. (b)κ_{pc_2}. (c)κ_{pc_3}.



(a) (b)

(c) (d)

Figure 4.64: Results of reconstructing an image from its eigenimages. (a) Original image adopted from [318]. (b) Using all eigenimages (PSNR=60dB). (c) ignoring one eigenimage (PSNR=38dB). (c) ignoring two eigenimages (PSNR=31dB).

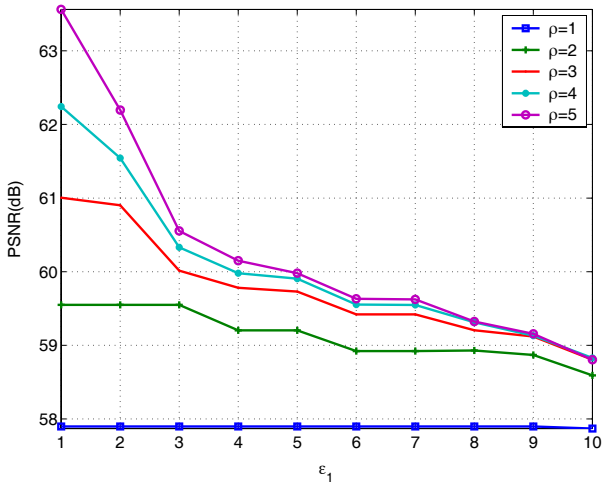
(a) (b)

(c) (d)

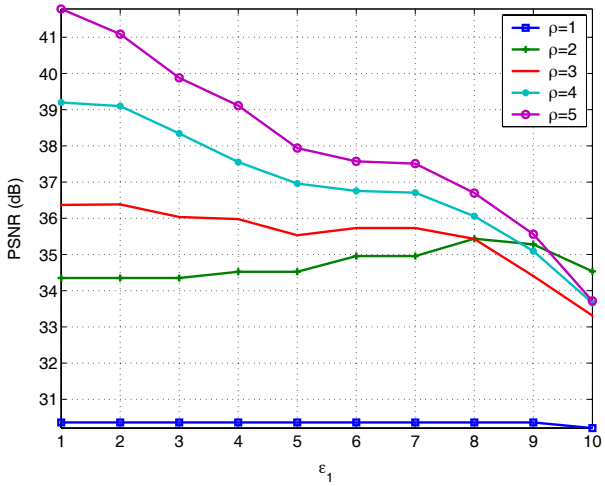
Figure 4.65: Results of reconstructing an image from its eigenimages. (a) Original image. (b) using all eigenimages (PSNR=62dB). (c) ignoring one eigenimage (PSNR=40dB). and (c) ignoring two eigenimages (PSNR=31dB).

of about $60dB$, while neglecting one and two eigenimages results in PSNR$\geq 35dB$ and PSNR$\geq 28dB$, respectively.

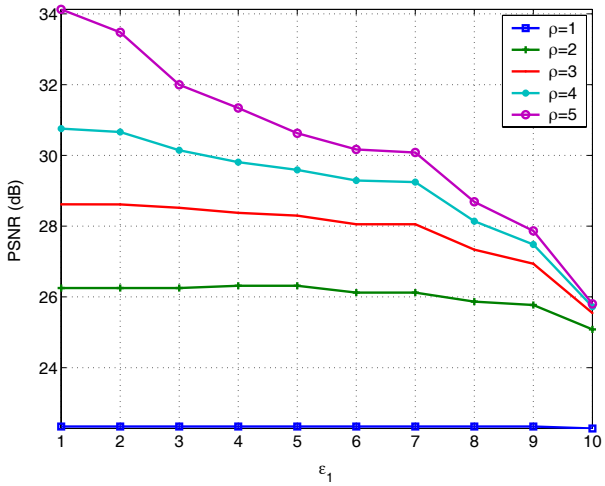
Figure 4.68 shows the results of the proposed compression method applied on the images shown in Figure 4.67. Table 4.13 shows the compression ratio used for compressing the eigenimages and the resulting compression ratio and PSNR values. These results are acquired while setting $p = \frac{1}{2}$, $\varepsilon_1 = 5$, and $\varrho = 5$. Figure 4.69 shows the exaggerated difference between the reconstructed images shown in Figure 4.68 and the original images shown in Figure 4.67. Here, the exaggeration scheme is defined as $x^\star = (x - [\eta - \sigma])/(2\sigma)$, where η and σ denote the expectation and the standard deviation of x, respectively. Note the high compression ratio of about $70 : 1$ in all cases, while the PSNR is mostly above $25dB$. Among other region–based coding approaches the method by *Carveic et. al.* is one of the best [84]. They mixed the color and texture information into a single vector and performed the coding using a massively computationally expensive algorithm. The final results show compression ratio of about $20 : 1$ for compression ratios of about $40dB$. In [104] the researchers use the same separation scheme between compression in the two disjoint domains of spectral and spatial redundancy using a PCA neural network. They reached the compression ratio of $3.7 : 1$ with value of PSNR around $25dB$, while almost all test samples are homogenous. In [400] the method gives the compression ratio of about $14.5 : 1$ but with the same range of PSNR as ours.



(a)

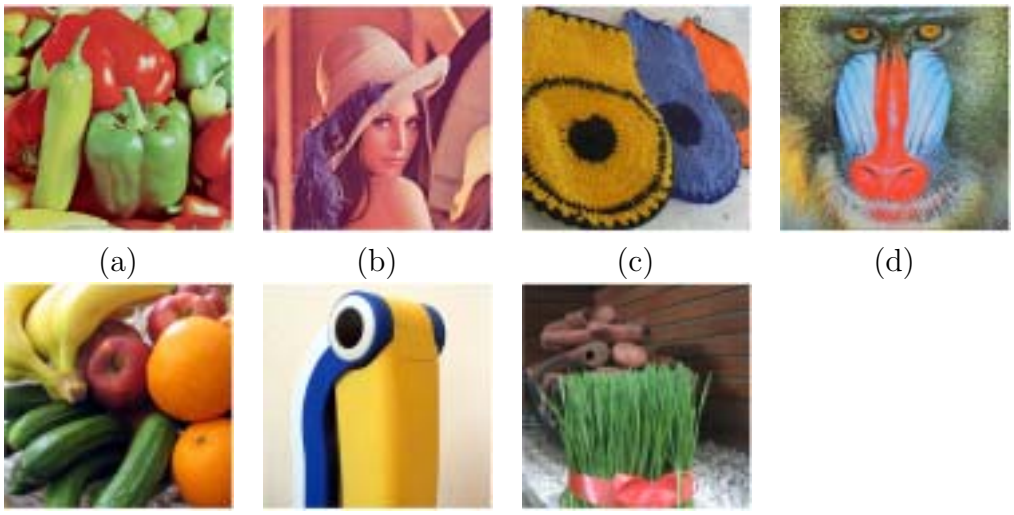


(b)



(c)

Figure 4.66: PSNR values of image reconstruction using. (a)All three eigenimages. (b)Only two eigenimages. (c)Just one eigenimage for different values of ε_1 and ϱ.



(a) (b) (c) (d)

(e) (f) (g)

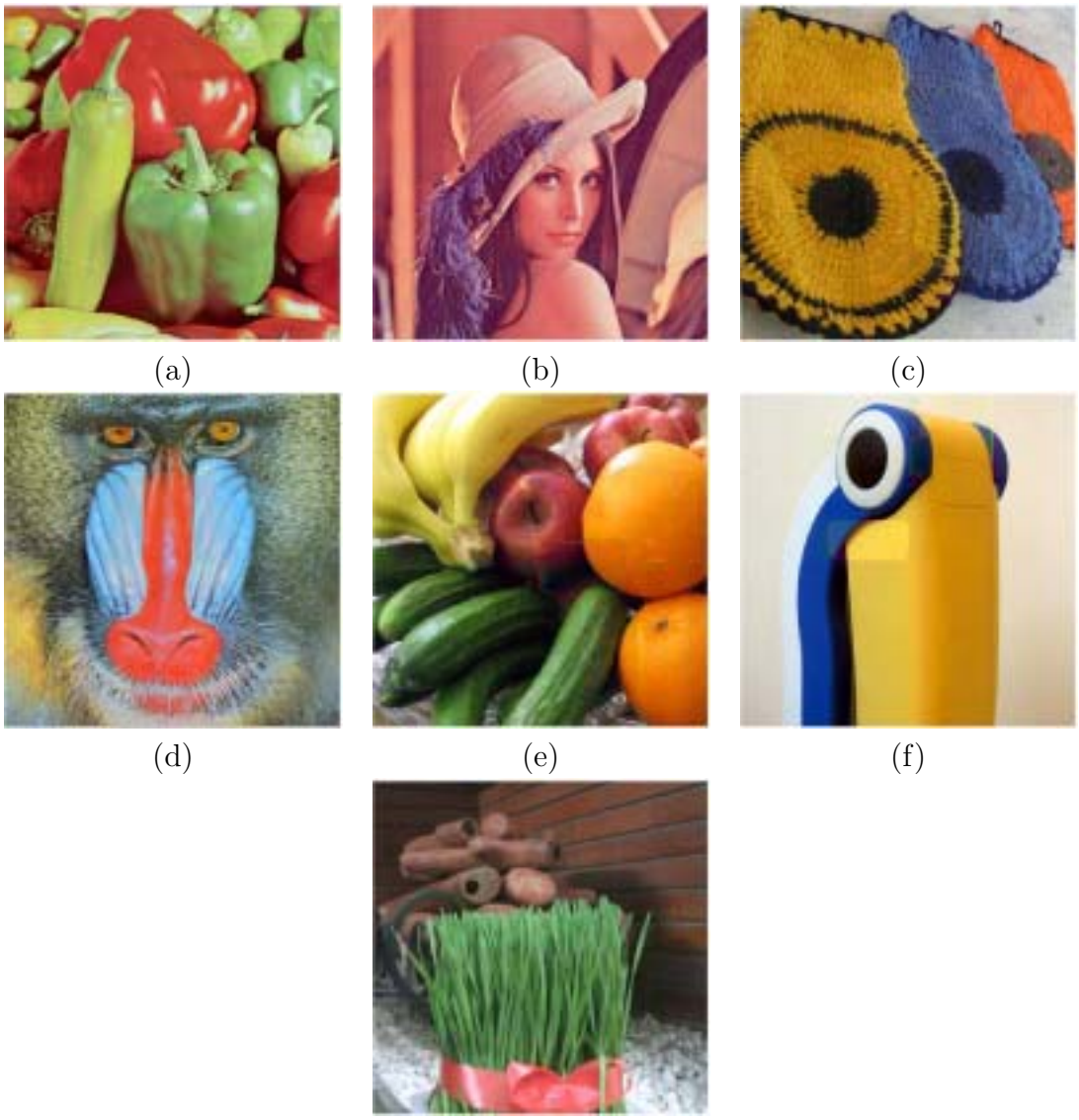
Figure 4.67: Some of the sample images used for testing the performance of the proposed compression method.

The only drawback of the proposed compression method is some rectangular artifacts in ultra simple images, as seen in Figure 4.68–f. The future plan is to overcome this minority of problem.

4.8 Watermarking

4.8.1 Block–Wise Interpolation

Figure 4.70–b shows the results of embedding the watermark data shown in Figure 4.70–a into the image shown in Figure 4.57–a with the resulting PSNR of $35dB$. Figure 4.70–c shows the exaggerated difference between the original image and the watermarked image, and Figure 4.70–d shows the extracted watermark data. Investigating Figure 4.70–c shows where the method hides the data; at each pixel, the



(a) (b) (c)

(d) (e) (f)

(g)

Figure 4.68: Results of applying the proposed compression method applied on the images shown in Figure 4.67. (for details see Table 4.13).

Table 4.13: Numerical results and parameters used for the results shown in Figure 4.68. [λ: Compression Ratio, bpp: bit per pixel, n: block count].

Sample	n	pc_1		pc_2		pc_3		**Image**		
		λ	bpp	λ	bpp	λ	bpp	λ	bpp	PSNR (dB)
4.67–a	334	42.6	0.19	58.8	0.14	∞	0	71.1	0.34	28.4
4.67–b	60	43.1	0.19	56.2	0.14	∞	0	72.6	0.33	33.6
4.67–c	161	39.8	0.20	55.1	0.15	∞	0	68.0	0.35	30.4
4.67–d	311	44.4	0.18	58.0	0.14	∞	0	72.5	0.33	24.5
4.67–e	203	41.3	0.19	56.9	0.14	∞	0	70.1	0.34	33.3
4.67–f	32	45.5	0.18	62.8	0.13	∞	0	78.8	0.30	26.1
4.67–g	91	41.3	0.19	57.7	0.14	∞	0	71.4	0.34	35.4

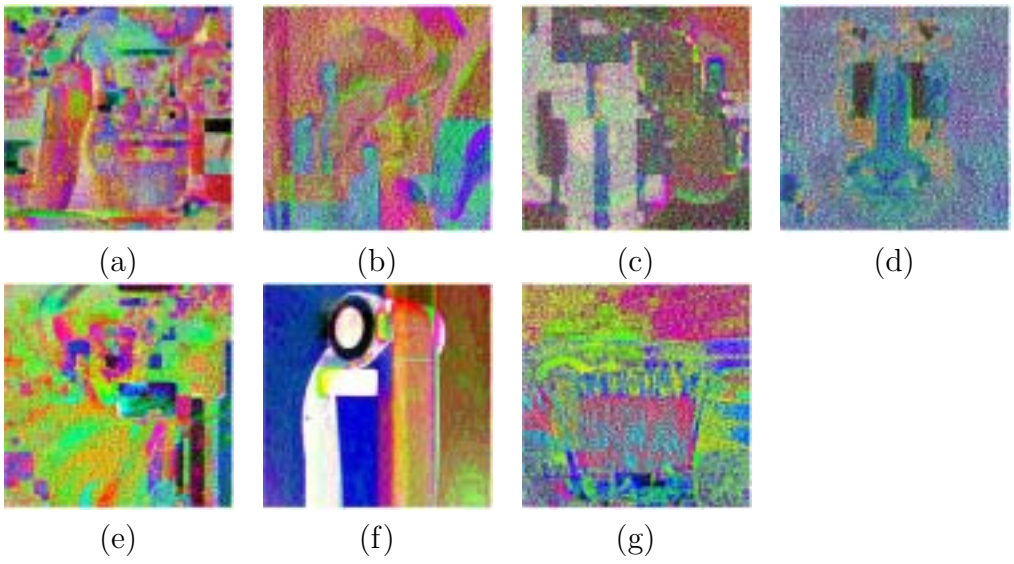
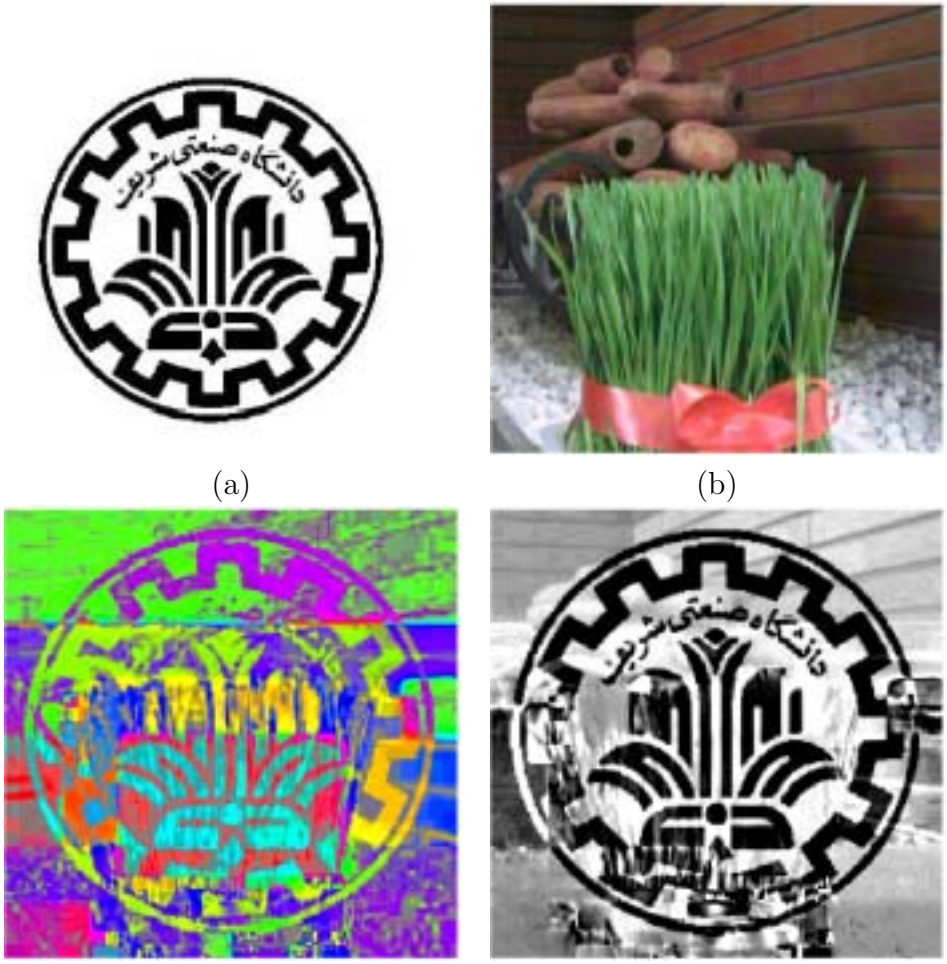


Figure 4.69: Exaggerated error of the proposed compression method for the images shown in Figure 4.67.

direction of the third principal component shows the direction in which data can be put such that does it not affect the visual appeal of the image. Figure 4.71 shows the same results for the image shown in Figure 4.57–b. Note that in both cases the difference image is highly exaggerated to give better visualization.

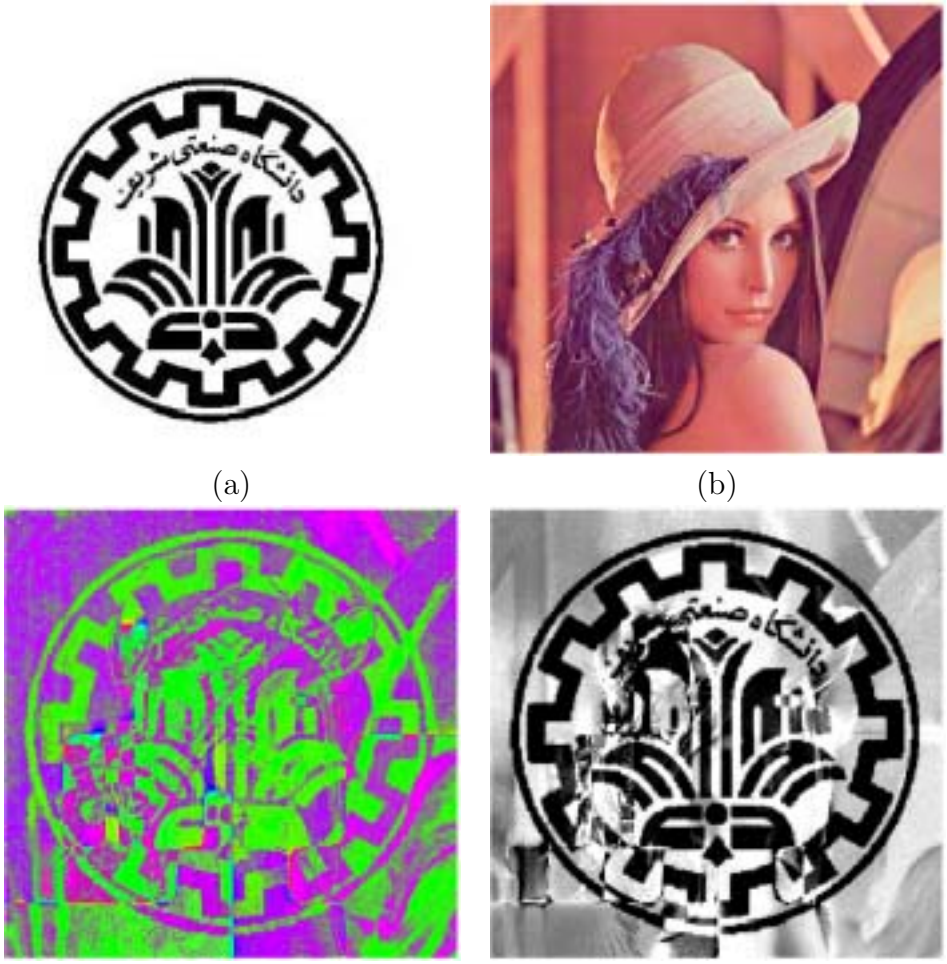
To investigate the robustness of the proposed watermarking method against invasive attacks, the 9 watermarks shown in Figure 4.72 were embedded to the sample images. Then, the watermarked images are attacked by using the *Adobe Photoshop 6.0*. Table 4.14 lists the description of some typical attacks. To illustrate the results the image shown in Figure 4.57–a with the watermark shown in Figure 4.72–c is shown. Figure 4.73 shows some of the attacked watermarked images and Figure 4.74 show the corresponding extracted watermark data. Investigating Figure 4.74 shows that the proposed watermarking method is robust against linear and nonlinear geometrical transformations, including rotation, scaling, trimming, and other geometrical distortions. Also, it is robust against occlusion, artistic effects, noise, enhancement operations like brightening and increasing contrast (even when performed locally), and also lossy compression ratios up to $15:1$. The reader is referred to the attacked images shown in Figure 4.73 and the corresponding extracted watermarks shown in Figure 4.74. Note that in all cases the method have no need of the original image thus, working in a complete blind condition.



(a) (b)

(c) (d)

Figure 4.70: Results of the proposed watermarking method performed on the image shown in Figure 4.57–a. (a) Watermark data. (b) Watermarked image with PSNR=35dB. (c) Exaggerated difference between the original image and the watermarked image. (d) Extracted watermark data.



(a) (b)

(c) (d)

Figure 4.71: Results of the proposed watermarking method performed on the image shown in Figure 4.58–a. (a)Watermark data. (b)Watermarked image with PSNR=37dB. (c) Exaggerated difference between the original image and the watermarked image. (d) Extracted watermark data.

Figure 4.72: Sample watermarks.

Table 4.14: Some of the attacks performed on the watermarked images to analyze the robustness of the proposed watermarking method.

index	**Attack**
0	–
1	Enhancement
2	Scaling and trimming
3	Rotation, scaling, and trimming
4–7	Nonlinear Geomtrical
8	Gaussian Lowpass
9,10	Scratch
11,12	Local brightening/darkening
13	Text
14	Motion Blur
15	Radial Blur
16–22	Geometrical Distortion
23	Uniform Color Noise
24	Gaussian Color Noise
25	Uniform Gray Noise
26	Gaussian Gray Noise
27	Median
28	Artistic Lens Flare
29	Lossy Compression

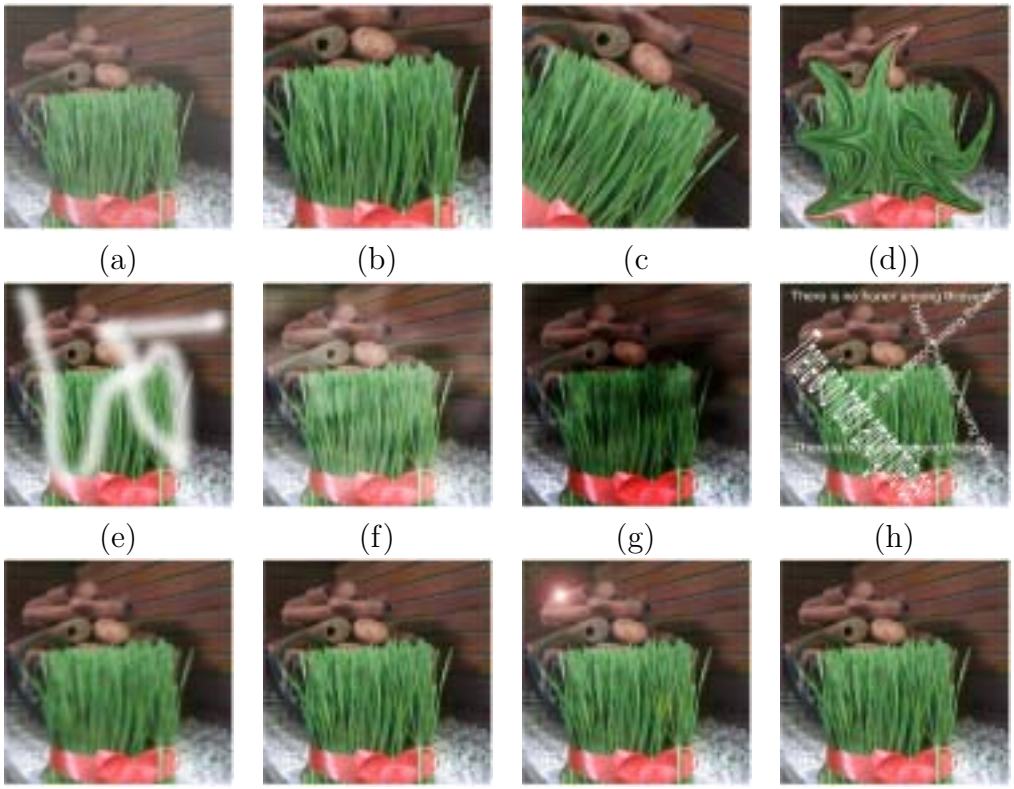


Figure 4.73: Some attacked watermarked images, corresponding items in Table 4.14 are: (a):1, (b):2, (c):3, (d):5, (e):9, (f):11, (g):12, (h):13, (i):14, (j):23, (k):28, and (l):29

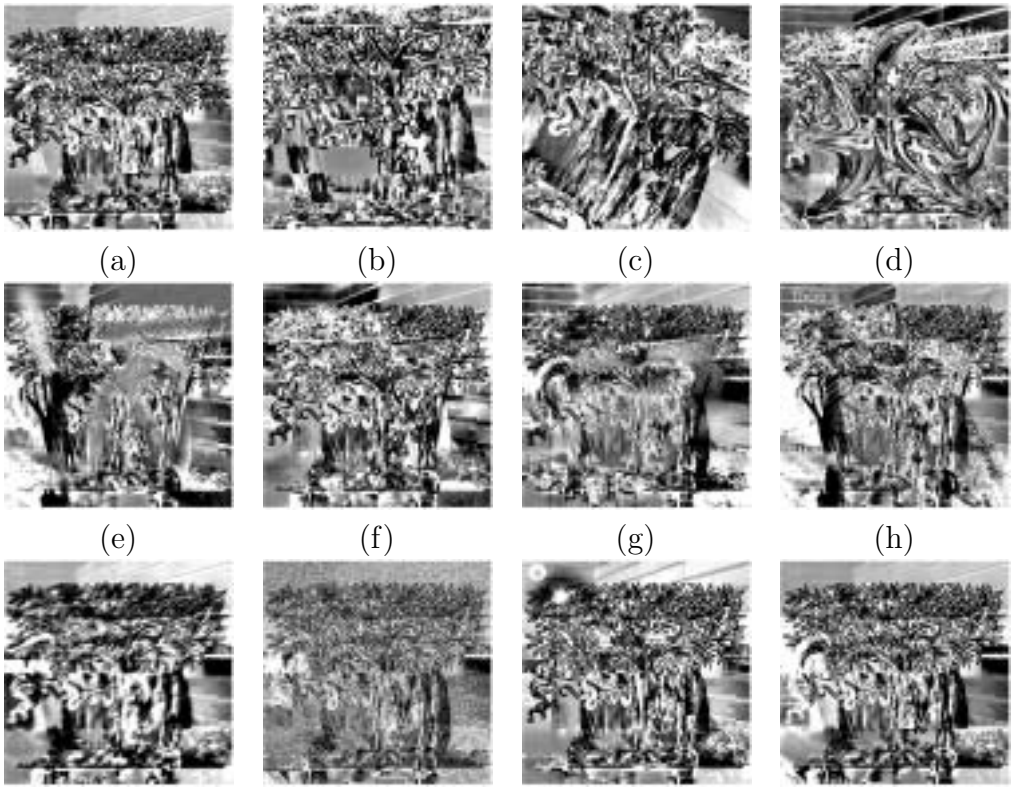


Figure 4.74: Extracted watermarks from attacked images of Figure 4.73.

Table 4.15 compares the proposed watermarking method with the available literature. The table lists the watermark capacity of each method when embedding to a 512×512 color image along with the domain in which data is embedded. Also, the attack resistance of different approaches is compared here. It is observed in different experiments that the standard deviation of pc_3 in a typical image is more than 4. Thus, using the proposed watermarking method on a 512×512 color image at least a same–sized $2bpp$ image can be used as the watermark signal. This makes the watermark capacity of the proposed method equal to 64KB. This is four times more than the highest capacity of the available approaches (the method by *Barni et. al.* [55]). The only approaches using the color vectors are proposed by *Chou et. al.* [101] and *Piyu et. al.* [360]. Note that the method by *Chou et al.* [101] is the only method showing resistance to the linear point operations like brightening and contrast enhancement. It must be emphasized that their method's resistance is limited to the global such operations, while the proposed method is resistant even to local linear point operations (see Figure 4.73–f,g). Unfortunately, no attention is spent on non–linear geometrical operations like elastic and perspective transformation and image editing processes like adding text, artistic effects, occlusion and so on. While many copyrighted images are used in books, posters, and websites where they appear with some levels of artistic manipulation, the non–efficiency of the available watermarking literature in dealing with these attacks is a real shortcoming. Table 4.15 depicts

Figure 4.75: Some sample images.

that the proposed method is one of the only available methods resistant to the seven groups of attacks listed in its caption.

4.8.2 Clustering–Based Approach

Figure 4.75 shows some of the test images used for analyzing the performance of the proposed method. These images are *Lena*, *Peppers*, and *Mandrill*. Figure 4.72 shows the sample watermarks used in this study.

Figure 4.76 shows the eigenimages extracted by the proposed method from the standard images shown in Figure 4.75 when setting $c = 5$. Each eigenimage is normalized to span the $[0, \cdots, 255]$ interval. To obtain a numerical analysis of the proposed eigenimage extraction method, the same test is performed on each sample image for each value of c in the $[1, \cdots, 10]$ interval. As the proposed clustering method contains some levels of non–repeatability, each experiment is performed ten times and the average values are used. Figure 4.77 shows the time needed to extract the eigenimages

(t_1) and the time elapsed when reconstructing the image from its eigenimages (t_2). Figure 4.77–a shows that, the time needed to extract the eigenimages of a given image have an almost quadratic relation to the number of clusters c ($t_1 \propto c^2$). Also, Figure 4.77–b demonstrates that reconstructing and image from its eigenimages elapses proportional to the number of clusters ($t_2 \propto c$). In the worst case ($c = 10$) computing the eigenimages lasts less than one minute while reconstructing the image takes less than five seconds. Note that the code is in MATLAB.

Figure 4.78 shows the values of κ_1, κ_2, and κ_3 in each image with each value of c. In this way, Figure 4.78 shows the compaction of the energy in the three eigenimages. The case of $c = 1$ corresponds to computing the eigenimages using the PCA in the entire image as a whole. This obvious case is about 10% worse than the cases produced by selecting values of c larger than unity. Note that selecting $c \neq 1$ switches the method from PCA to local PCA. In all cases, selecting values of c larger than 5 shows negligible change in the energy spread pattern between eigenimages, while almost tripling the elapsed time. This experiment along with others performed on other images proves that $c = 5$ is an ideal compromise.

In this thesis, we use the eigenimages to embed data in a color image. Thus, it is essential to estimate the ratio of information lost when neglecting an eigenimage or replacing it with the watermark signal. Figure 4.79 shows the values of PSNR when reconstructing the sample images while neglecting the third eigenimage. Figure 4.80

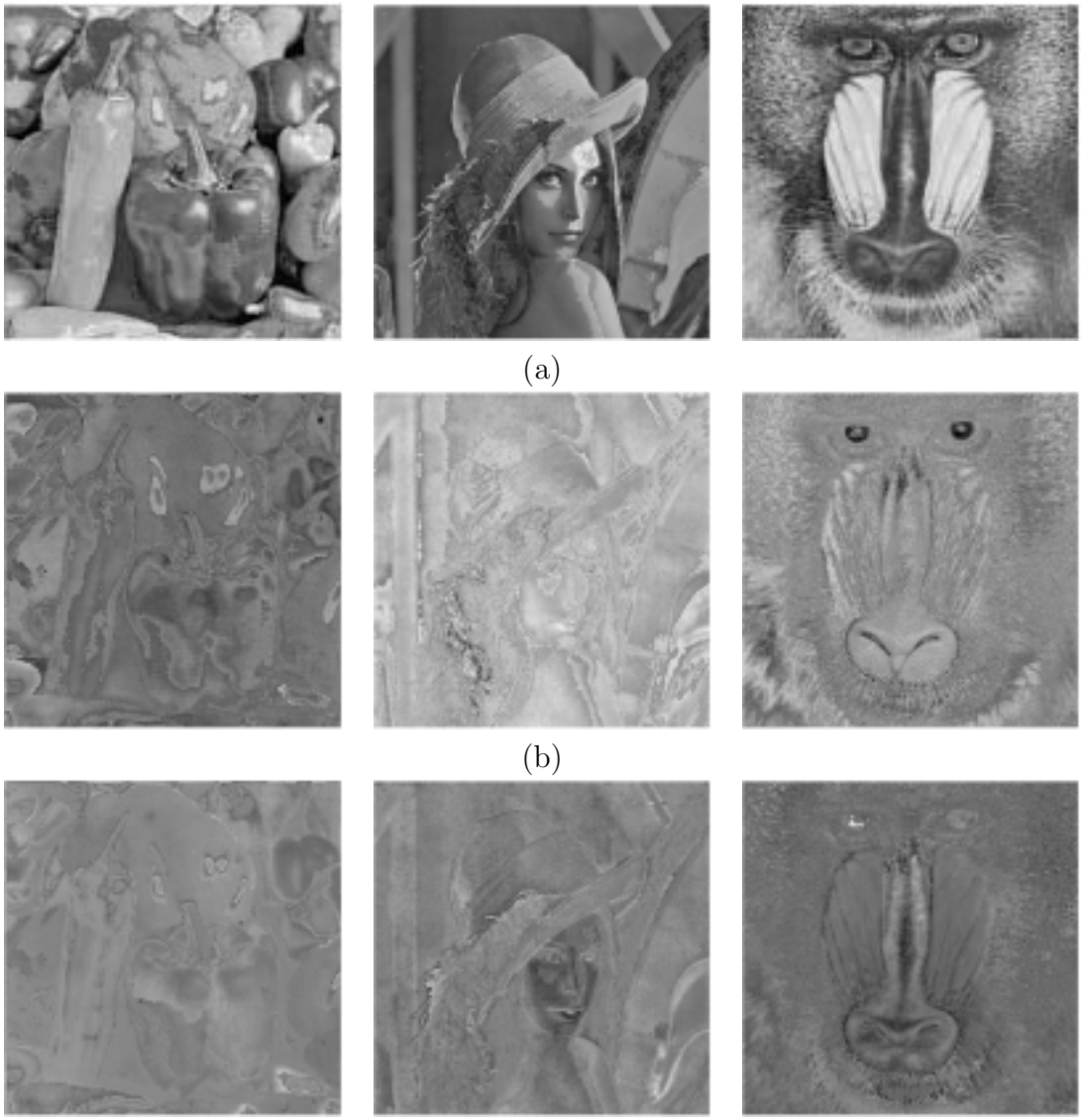
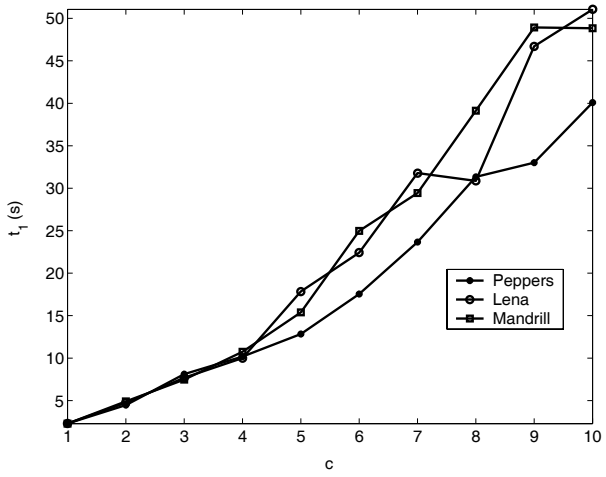
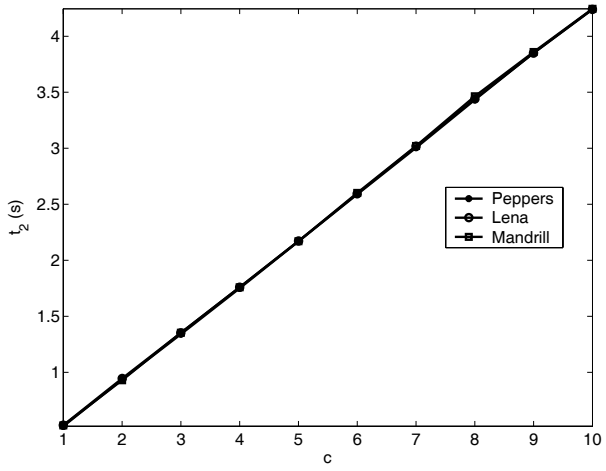


Figure 4.76: Eigenimages extracted from the sample images shown in Figure 4.75 using the proposed method. Dynamic range of the images is exaggerated to give a better visual appearance. (a) First eigenimage. (b) Second eigenimage. (c) Third eigenimage. Number of clusters is set to 5.

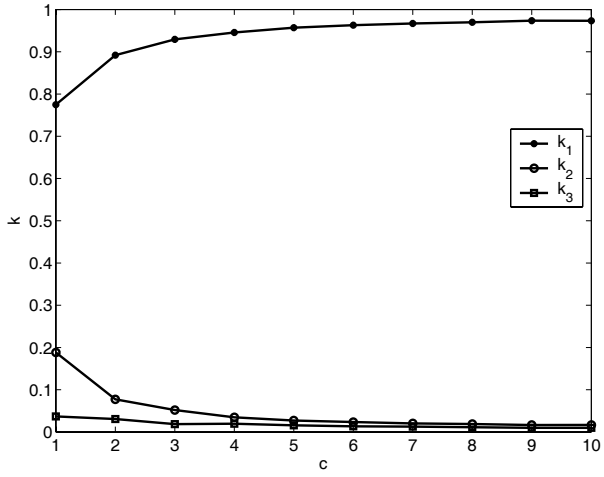


(a)

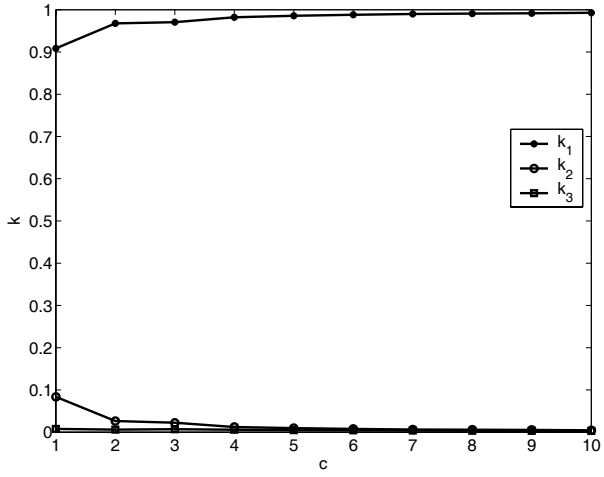


(b)

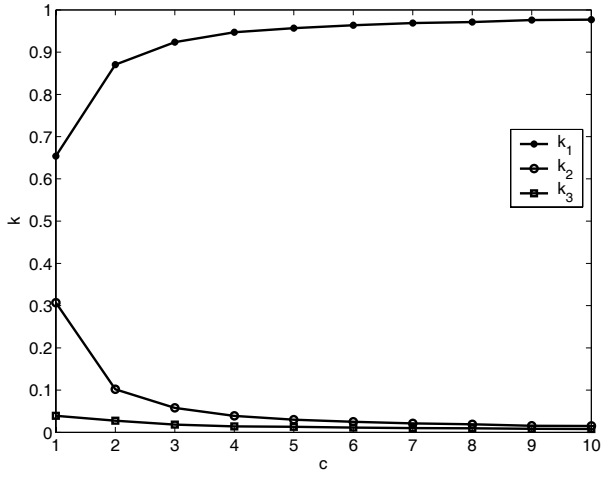
Figure 4.77: Elapsed times by the proposed method. (a) Extracting the eigenimages. (b) Reconstructing the image.



(a)



(b)



(c)

Figure 4.78: Energy compaction of the eigenimages. (a) *Peppers*. (b) *Lena*. (c) *Mandrill*.

shows the PSNR values when replacing the third eigenimage with the watermark data (watermarking the image). In the watermarking test the watermark image shown in Figure 4.72–a is used. Figure 4.79 shows that neglecting the third eigenimage results in PSNR value of more than $25dB$. Also, the choice of $c = 5$ is reasonable here, because the main rise of the PSNR curve happens when $c \leq 5$. Comparing Figure 4.80 to Figure 4.79 shows that embedding the watermark results in an averagely $2dB$ decline in the PSNR, while again the PSNR is higher than $25dB$ in this test. In other tests the same range of PSNR is observed but sometimes the watermarked image shows higher PSNR when compared to the image produced by neglecting the third eigenimage. Figure 4.81 shows the results of neglecting the third eigenimage and Figure 4.82 illustrates the case of replacing it with the watermark data shown in Figure 4.72–b. In these tests the value of c equal to 5 is used. Figure 4.83 shows the extracted watermarks from the images shown in Figure 4.82 and Figure 4.84 shows the exaggerated difference between the images shown in Figure 4.82 and the original images shown in Figure 4.75. Investigating Figure 4.84 shows where the method embeds the data; at each pixel, the direction of the third principal component shows the direction in which data can be placed while not affecting the visual appeal of the image.

An experiment is conducted to test the sensitivity of the method to the proper selection of c. The watermark signal is embedded into the sample images using $c = c_1$

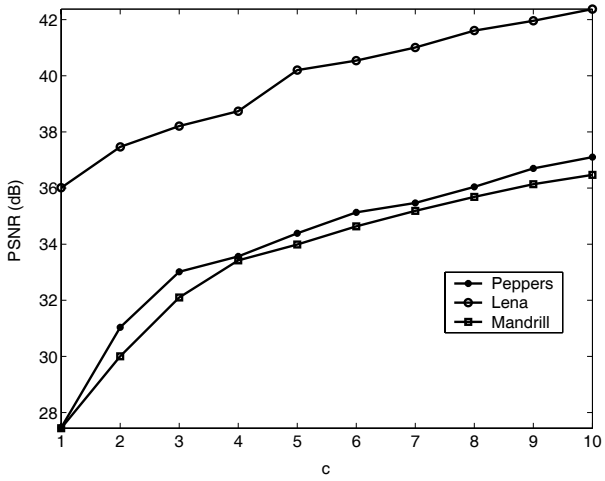


Figure 4.79: PSNR values for different number of clusters when neglecting the third eigenimage.

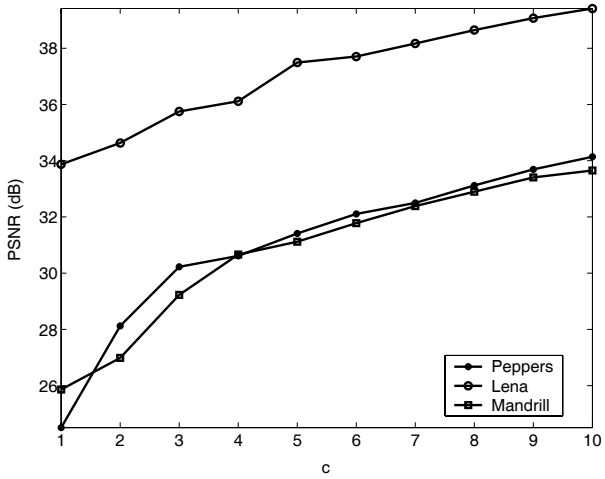


Figure 4.80: PSNR values for different number of clusters after watermarking.

Figure 4.81: Results of neglecting the third eigenimage.

Figure 4.82: Results of the proposed watermarking method.

Figure 4.83: Watermark signals extracted from images shown in Figure 4.82.

Figure 4.84: Exaggerated difference between original images shown in Figure 4.72 and watermarked images shown in Figure 4.83.

and then the extraction process is performed using values of $c = c_2 \neq c_1$. Figure 4.85 shows a typical set of results when $c_1 = 5$ and $c_2 = 1, \cdots 9$. The embedded watermark is distinguishable in all cases. Thus, there is no need to know the exact value of c used for embedding the watermark to an image to extract the watermark from it.

To investigate the robustness of the proposed watermarking method against invasive attacks, the watermarked images are attacked by some methods using the *Adobe Photoshop 6.0*. Figure 4.86 shows some of the attacked watermarked images and Figure 4.87 shows the corresponding extracted watermarks. The attacks include rotation, scaling, translation, adding text, blurring, artistic effects (plastic wrap and lens flare), adding noise (PSNR=$25dB$), geometrical distortion, JPEG compression ($20:1$), and color distortion. The watermark image is highly distinguishable in all samples shown in Figure 4.87. Investigating Figure 4.87 along with the other numerous tests shows that the proposed watermarking method is robust against linear

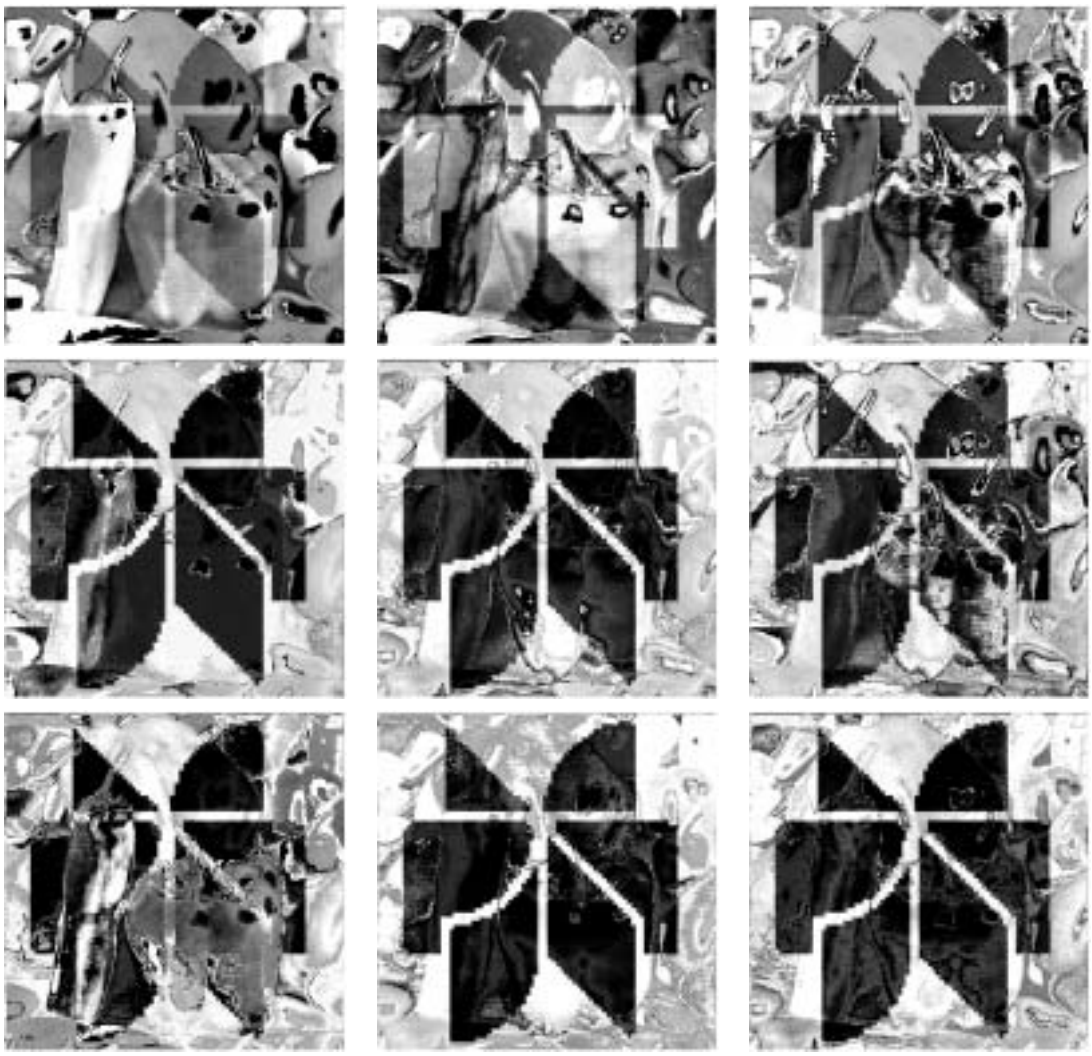


Figure 4.85: Results of extracting the watermark using values of $c = 1, \cdots 9$. The actual watermarking is performed using $c = 5$.

and nonlinear geometrical transformations including rotation, scaling, cropping, and other geometrical distortions. Also, it is robust against occlusion, artistic effects, captioning, noise addition, enhancement operations like brightening and increasing contrast (even when performed locally), lossy compression (up to 20 : 1), frequency domain filtering, and different kinds of blurring.

Table 4.15 compares the performance of the proposed watermarking method with the best methods available in the literature. The table lists the watermark capacity of each method when embedding to a 512×512 color image along with the domain in which data is embedded. Also, the results of the attack resistance of different approaches is compared in that table. It is observed in different experiments that the standard deviation of pc_3 in a typical image is more than 8. Thus, using the proposed watermarking method on a 512×512 color image at least a same–sized $2bpp$ image can be used as the watermark signal (2 comes from $\log_2 \frac{8}{2}$). This makes the watermark capacity of the proposed method equal to 64KB. This is four times more than the highest capacity of the available approaches (the method by *Barni et. al.* [55]). The only approaches using the color vectors are proposed by *Chou et. al.* [101] and *Piyu et. al.* [360]. Note that, the method by *Chou et. al.* [101] is the only method showing resistance to the linear point operations like brightening and contrast enhancement. It must be emphasized that their method's resistance is limited to the global such operations, while the proposed method is resistant even to local linear

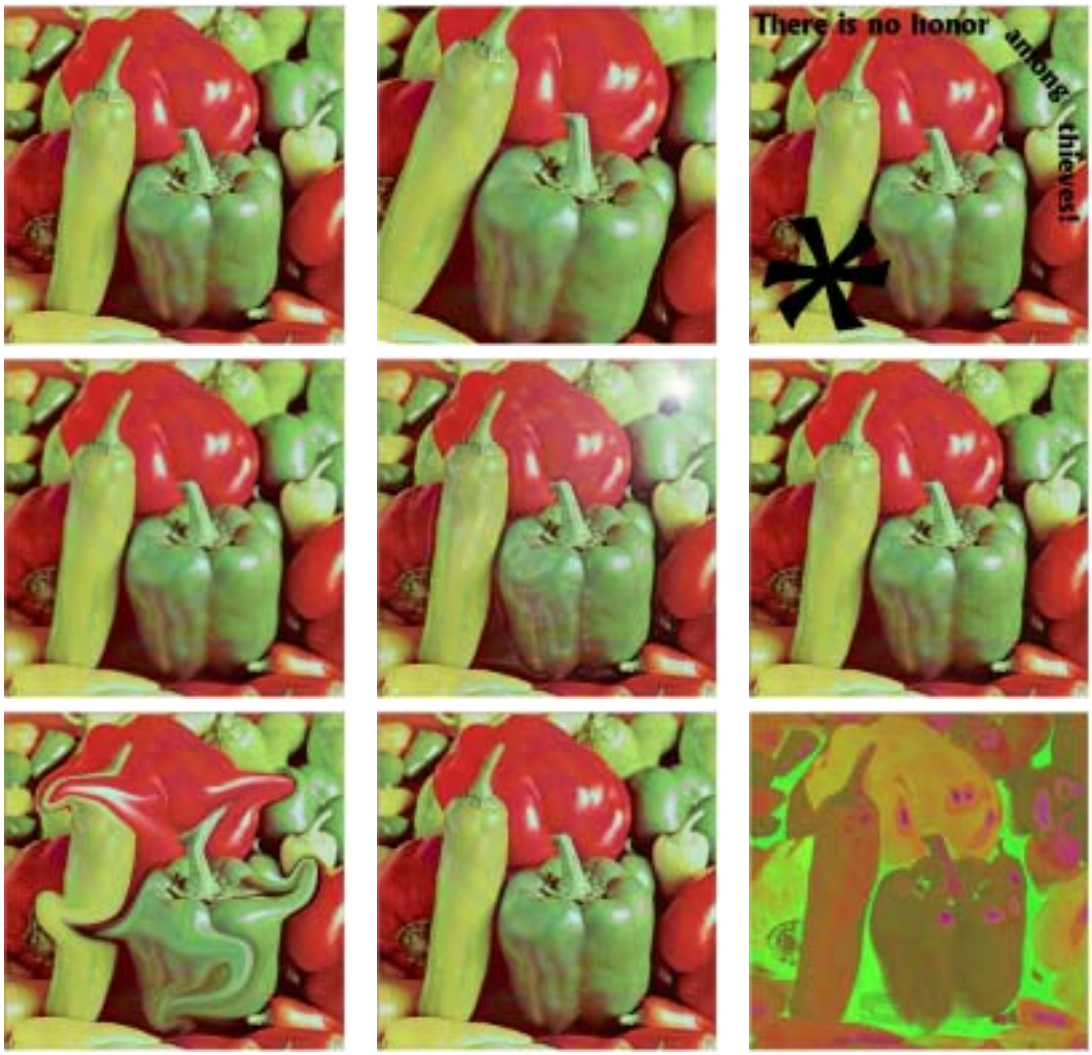


Figure 4.86: Attacked watermarked images.

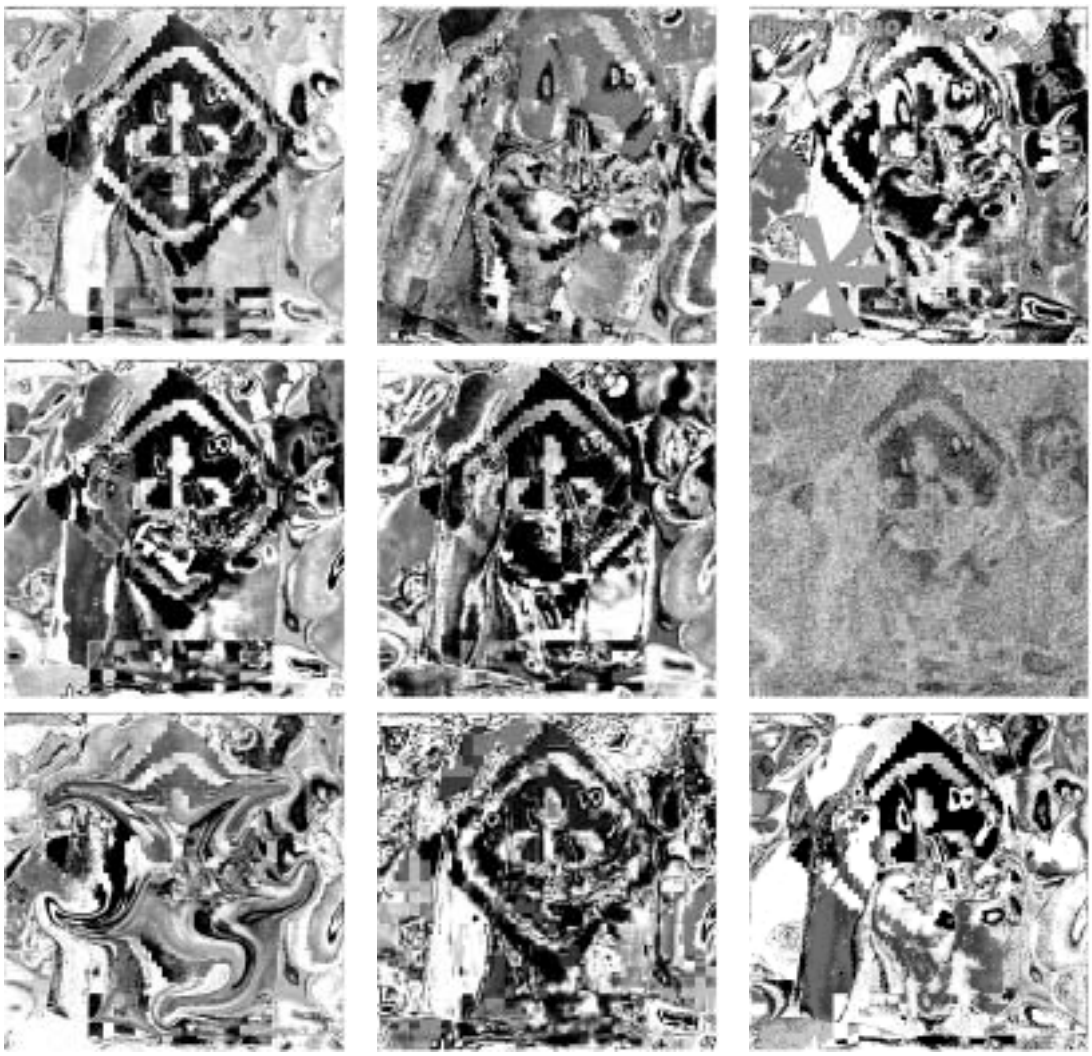


Figure 4.87: Extracted watermarks from the attacked watermarked images shown in Figure 4.86.

point operations. Unfortunately, no attention is spent on non–linear geometrical operations like elastic and perspective transformation and image editing processes like adding text, artistic effects, occlusion and so on. While many copyrighted images are used in books, posters, and websites where they appear with some levels of artistic manipulation, the non–efficiency of the available watermarking literature in dealing with these attacks is a real shortcoming. Table 4.15 depicts that the proposed method is the only available method resistant to the seven groups of attacks listed in its caption.

Here, we compare the proposed method with the method discussed in Section 4.8.1. Although, the method in Section 4.8.1 shows the same robustness as the proposed method, it elapses about 10 minutes on embedding or extracting the watermark to/from an image (compared to the less than one minute cost of the proposed method). Excluding the work in Section 4.8.1, we perform a comparison between the watermarking methods listed in Table 4.15. The only approaches using color vectors are proposed by *Chou et. al.* [101] and *Piyu et. al.* [360]. Note that, the method by *Chou et. al.* [101] is the only method showing resistance to the linear point operations like brightening and contrast enhancement. It must be emphasized that the attack resistance of the method in [101] is limited to the global operations, while the proposed method is resistant even to local linear point operations. Unfortunately, in the available literature, no attention is spent on non–linear geometrical operations

Table 4.15: Comparison of different watermarking methods with the proposed method used in a 512×512 image. –: Not Resistant. $\sim$: Partially Resistant. $\surd$: Completely Resistant. [Abbreviations: Cap: Capacity, G: Grayscale, SCC: Single Color Component, CV: Color Vector, LG: Linear Geometrical Transformation, NLG: Nonlinear Geometrical Transformation, LPO: Linear Point Operations, NLPO:Nonlinear Point Operations, SO: Spatial Domain Operations, EO: Editing Operations, CMP: JPEG Compression].

Method	**Cap.**	**Domain**	**Attack Resistance**						
			LG	NLG	LPO	NLPO	SO	EO	CMP
[304]	$4KB$	G	$\sim$	–	–	–	–	–	$\surd$
[267]	$8KB$	G	$\surd$	–	–	$\sim$	$\sim$	–	$\surd$
[177]	$8KB$	G	$\sim$	–	–	$\sim$	$\sim$	–	$\surd$
[101]	$2KB$	CV	$\sim$	–	$\sim$	$\sim$	$\sim$	–	$\surd$
[399]	$1KB$	SCC	$\sim$	–	–	$\sim$	$\sim$	–	$\surd$
[55]	$16KB$	SCC	$\surd$	–	–	$\sim$	$\sim$	–	$\surd$
[231]	$8B$	SCC	–	–	–	–	–	–	$\surd$
[347]	$0.5KB$	G	$\surd$	–	–	$\sim$	$\sim$	–	$\surd$
[203]	$60B$	G	$\surd$	–	–	–	$\sim$	–	$\surd$
[322]	$64B$	G	$\sim$	–	–	–	$\sim$	–	$\surd$
[360]	$2KB$	CV	$\sim$	–	–	$\sim$	$\sim$	–	$\surd$
Section 4.8.1	$64KB$	CV	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Section 4.8.2	$64KB$	CV	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

like elastic and perspective transformation and image editing processes like adding text, artistic effects, occlusion and so on. While many copyrighted images are used in books, posters, and websites where they appear with some levels of artistic manipulations, the non–efficiency of the available watermarking literature in dealing with these attacks is a real shortcoming. Table 4.15 depicts that the proposed method is one of the only available method resistant to the attacks listed in its caption.

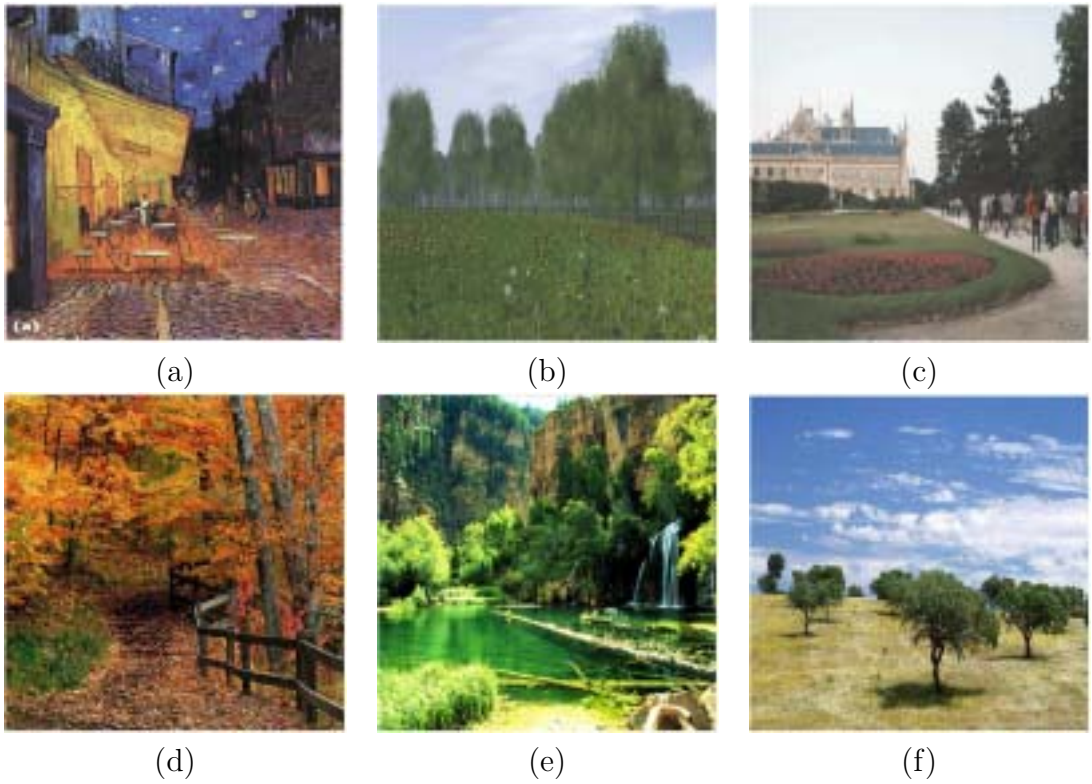
4.9 Direct Copy Protection

4.9.1 Clustering–Based Approach

Figure 3.75 shows the homogenous swatch bank used in this work. The bank contains 49 swatches manually extracted from some color images captured by an *A60 Canon* digital camera. The images are acquired from the natural and artificial objects in daylight or with flash. No other specific constraints are fulfilled. The color descriptors of the swatches of the bank are saved in a *.mat* file occupying $6,184$ bytes of memory (in non–compressed format). After this stage, the swatches are not used and the extracted color information are used.

Figure 4.88 shows some sample images used in this study. All these images are adopted from color transfer literature. In different experiments, it is proved that the actual number of clusters of the FPCAC is not affecting the performance of the algorithm. In this thesis, we illustrate samples relating to the case of $C = 3$, but similar results are obtained for other values of C. In fact, no user supervision is needed for determining the actual value of C. Note that the actual value of C is a secured parameter, which blocks cheating attempts. The radius of the convolution kernel in all experiments is set to 5. There is no other parameters in the proposed method to be selected by the user. Hence, the method is a fully unsupervised approach.

The proposed encryption/decryption process is performed on images shown in Figure 4.88. The PSNR is computed between the original images and the decrypted



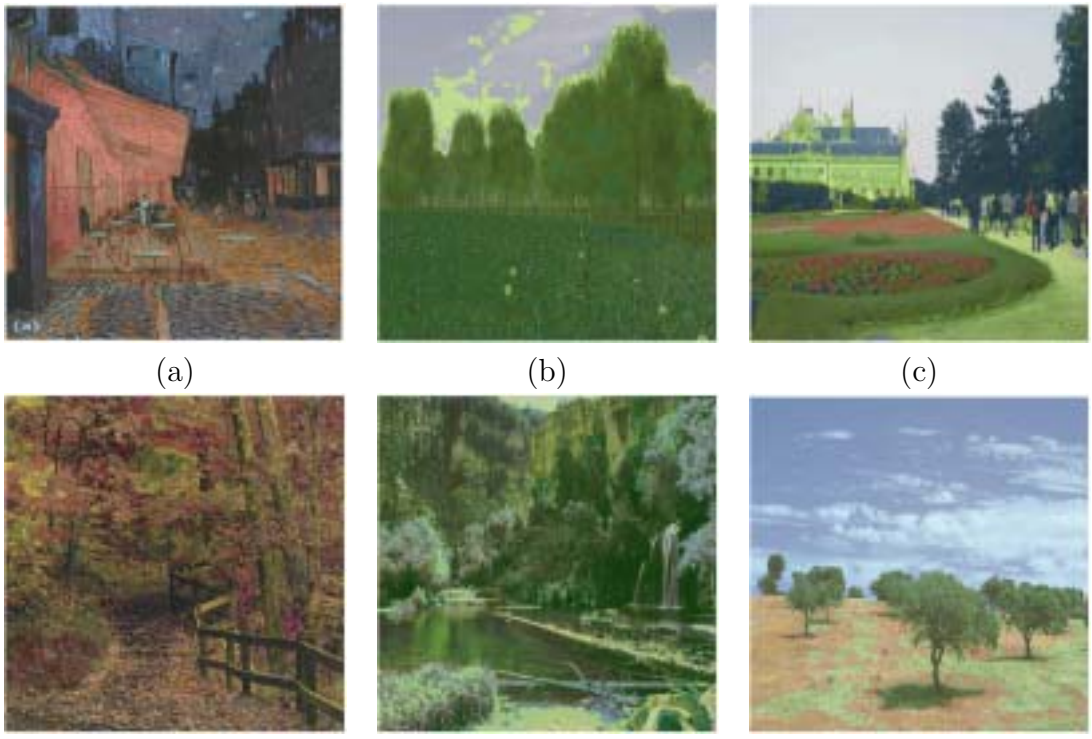
(a) (b) (c)

(d) (e) (f)

Figure 4.88: Some sample images. (a) *Vincent van Gogh*s *Cafe Terrace on the Place du Forum, Arles, at Night*, *Arles*, September 1888, oil on canvas, image from the Vincent van Gogh Gallery, *www.vangoghgallery.com*. (b) and (c) Courtesy of *Oliver Deussen*. (a), (b), and (c) Adopted from [297]. (d) *Mc. Cormic Creek State Park, Indiana* by *Mike Briner*, *mbphoto@spraynet.com*, *www.mikebrinerphoto.com*. (e) *Hanging Lake* by *Brent Reed*, *brent@reedservices.com*. (f) Adopted from [92].

images. Also, the redundancy of the key is computed. Two elapsed times are measured, t_1 is the time needed to encrypt an image and t_2 is the time needed to decrypt it. Numerical information are listed in Table 4.16. In this experiment, we have fixed the value of m equal to 1.05. It is observed in this experiment that in all cases, the PSNR is above $45dB$. Note that, as indicated recently, PSNR values of above $38dB$ are visually satisfactory even for professionals [207]. Figure 4.89 and Figure 4.90 show the encrypted and decrypted images, respectively. Note the distorted pattern of edges in the pavement, sky, and grass regions in Figures 4.89–a, 4.89–b and 4.89–c. Also, note that in all encrypted images while the contents of the image is readable, it is made unrealistic and unsatisfactory for unauthorized distribution. Also, note the perfect quality of the reconstructed images in Figure 4.90. None of the intentionally produced artifacts are visible in reconstructed images. Also, the redundancy of the key is less than 2% in all cases and the elapsed time for a set of encryption and decryption stages is $15s$ averagely (always less than $20s$). The interesting feature is that, averagely, 75% of the computational load is for the supplier, which can have specialized acceleration tools to boost the performance.

Figure 4.91 illustrates the sequential results of the proposed method. In this figure, Figure 4.91–a shows the original image, Figure 4.91–b shows the results of FPCA clustering, and Figure 4.91–c shows the encrypted image. Then, Figure 4.91–d shows the results of decryption. In this example, $m = 1.29$ and we have $t_1 = 7.7s$



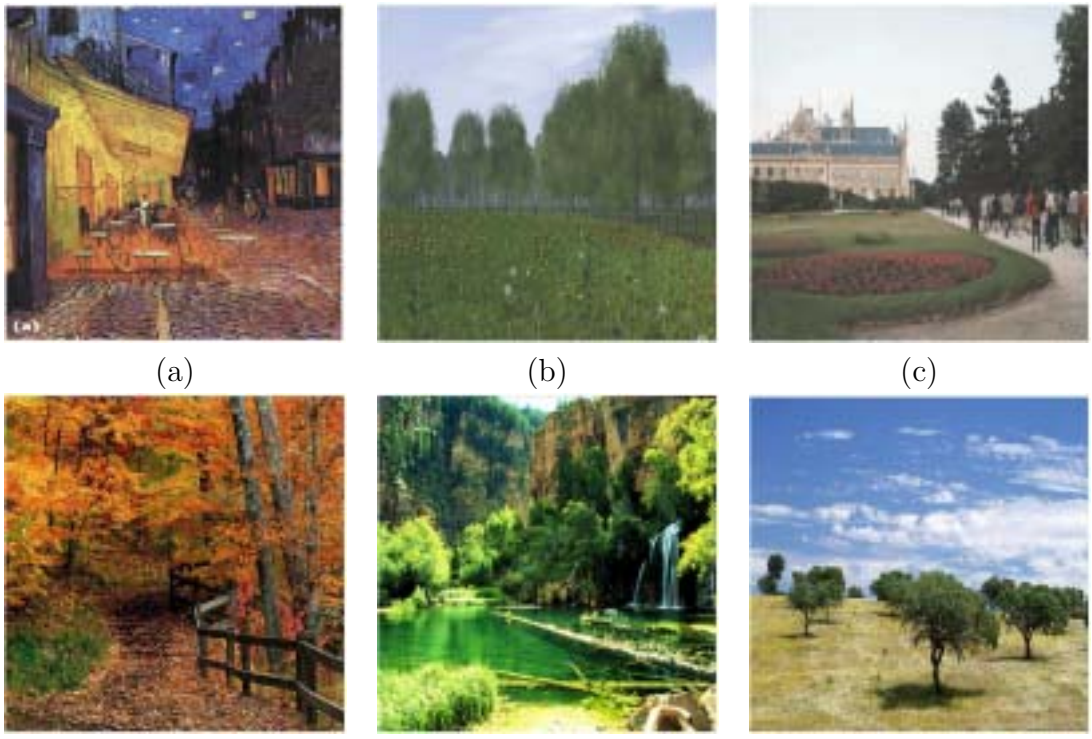
(a) (b) (c)

(d) (e) (f)

Figure 4.89: Encrypted versions of the images shown in Figure 4.88, using proposed algorithm.

Table 4.16: Numerical results of proposed algorithm performed on images shown in Figure 4.88. [m: Fuzziness. t_1 Encryption time. t_2 Decryption time. PSNR after reconstructing the image. ε Redundancy of key.]

Image	**Size**	m	t_1	t_2	PSNR	ε
4.88–a	482×683	1.05	$8.9s$	$2.8s$	$45.3dB$	0.81%
4.88–b	896×602	1.05	$14.9s$	$5.2s$	$46.6dB$	0.60%
4.88–c	896×602	1.05	$16.4s$	$5.1s$	$46.9dB$	0.73%
4.88–d	600×450	1.05	$8.2s$	$2.6s$	$45.7dB$	1.69%
4.88–e	600×450	1.05	$10.8s$	$2.7s$	$45.0dB$	1.30%
4.88–f	680×448	1.05	$8.8s$	$3.0s$	$45.2dB$	0.73%



(a) (b) (c)

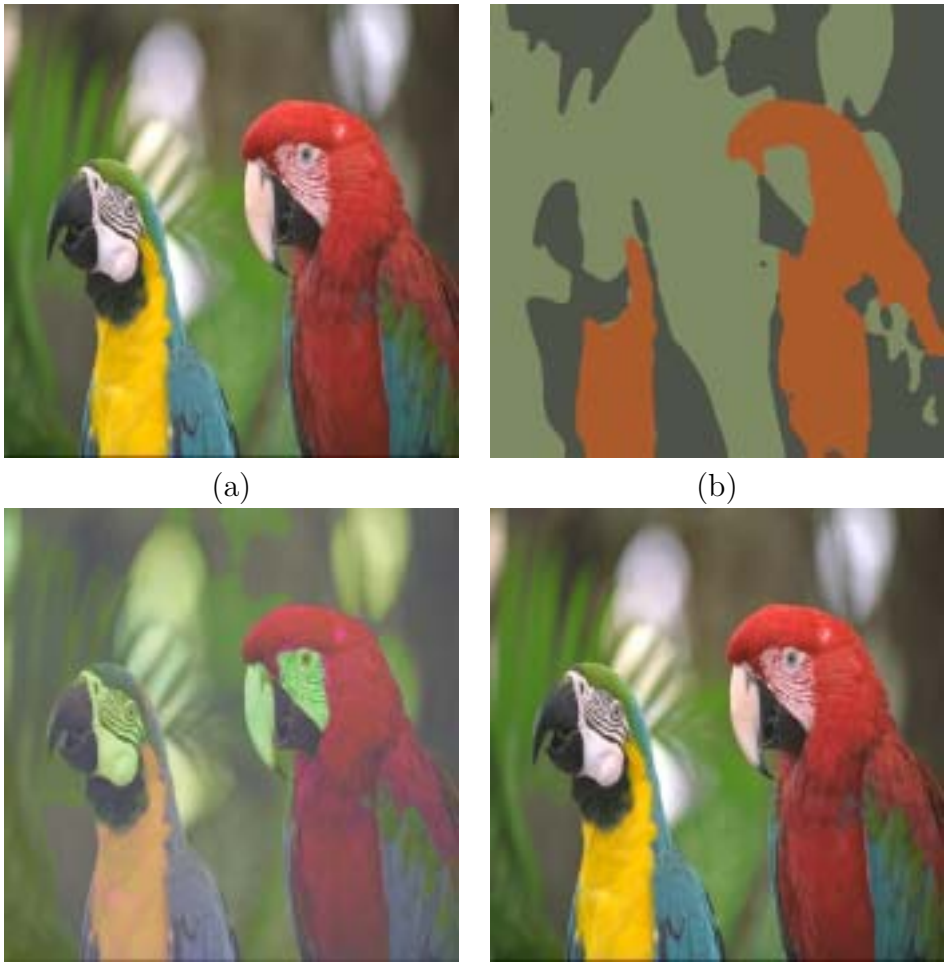
(d) (e) (f)

Figure 4.90: Decrypted versions of the images shown in Figure 4.89, using proposed method.

and $t_2 = 1.7s$. The resulting PSNR is $45dB$ and the redundancy of the key is less than 0.9%.

The important question here is the security of the proposed method. It is not enough to know m to infer the code of an image. This is based on an interesting feature of the FPCA clustering method. In fact, FPCAC is not a repeatable operation. This is theoretically trivial because FPCAC starts from an entirely random set of clusters. Figure 4.92 shows the results of 9 runs of FPCAC on a single image with a unique set of parameters ($C = 3, m = 1.2$). Table 4.17 lists the mean square distance between different clustering results. This table shows that there is averagely 16% difference between the clustering results of the same image, compensated for switched clusters. Another interesting feature is that only three pairs are more than 95% alike (a–e, b–i, and f–g). As such, in 9 trials, 6 distinct clustering results are encountered. We emphasize that here we have assumed that the attacker knows C and m, lacking knowledge of which makes cracking the code ultimately harder. Figure 4.93 shows 9 different encryptions of a single image with different values of m.

Here, we examine a scenario for attacking the proposed encryption method. Assume that the attacker has an encrypted image and a key to a different encryption of that image, and wants to use the key in place. In the worst situation, assume that the two encryptions share the same values of C and m. Figure 4.94 illustrates this situation. With the desperate results of this experiment (switched code resulted



(a) (b)

(c) (d)

Figure 4.91: Sequential results of proposed algorithm. (a) Original image, courtesy of Kodak *PhotoCD* (PCD0992). (b) Results of FPCA clustering. (c) Encrypted image. (d) Decrypted image.

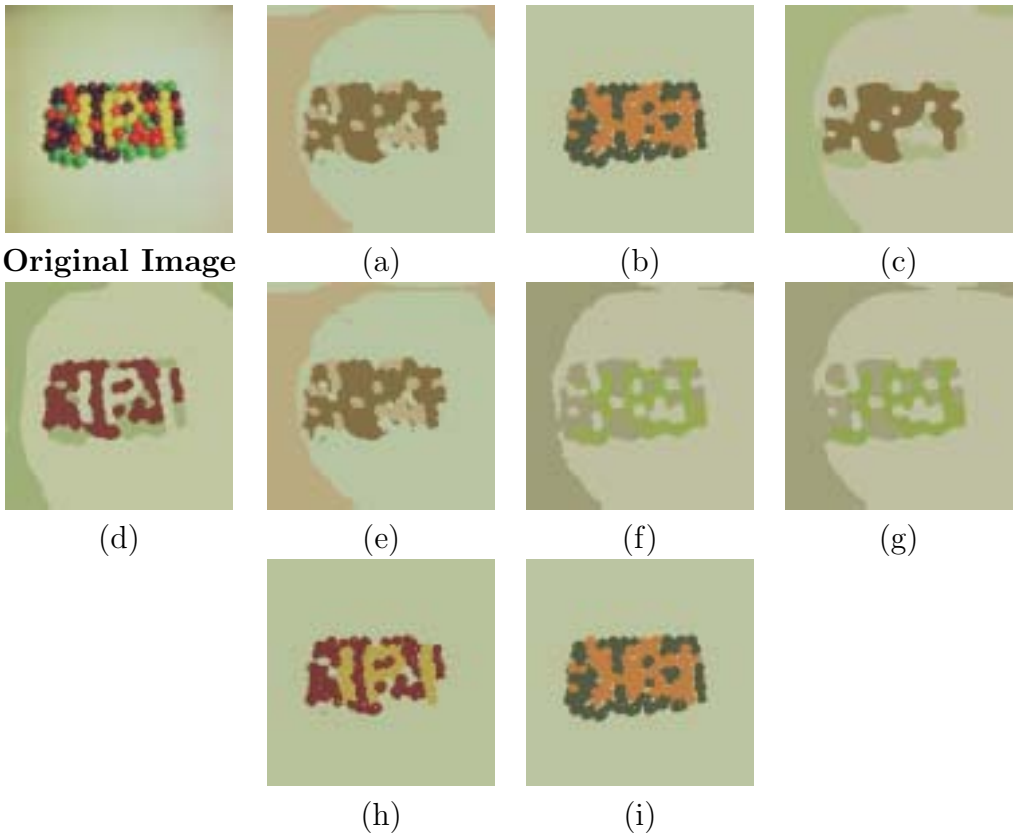


Figure 4.92: Results of applying FPCAC with a unique set of parameters on a single image for 9 different runs. Original image, courtesy of Signal and Image Processing Institute at University of Southern California.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 4.93: Different encryptions of a single image.

Table 4.17: Mean square distance between different clustering results of a single image with a unique set of parameters. Values less than 0.05 are written in bold. Images are shown in Figure 4.92.

Trial	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
(a)	0	0.23	0.091	0.17	**0.004**	0.14	0.12	0.21	0.23
(b)	0.23	0	0.23	0.17	0.23	0.21	0.23	0.059	**0.0006**
(c)	0.091	0.23	0	0.12	0.093	0.086	0.088	0.25	0.23
(d)	0.17	0.17	0.12	0	0.17	0.19	0.2	0.17	0.17
(e)	**0.004**	0.23	0.093	0.17	0	0.14	0.13	0.21	0.23
(f)	0.14	0.21	0.086	0.19	0.14	0	**0.024**	0.22	0.21
(g)	0.12	0.23	0.088	0.2	0.13	**0.024**	0	0.23	0.23
(h)	0.21	0.059	0.25	0.17	0.21	0.22	0.23	0	0.059
(i)	0.23	**0.0006**	0.23	0.17	0.23	0.21	0.23	0.059	0

in $PSNR < 12dB$), the high security of the proposed method is clear. The reader should be aware that the difference between two distinct encryption of a single image mainly relies on the different results of the FPCAC. Hence, in a practical system, having the history of encryption will help to avoid producing a similarly–repeated encryption. This will eventually result in maximum security of the proposed method.

We have not discussed the cracking attempt which depends on cheating the index map and the color descriptors manually. We believe that according to the complexity of producing an image–sized indexed map, this is a dead end. Also, note that as the edges of the index map are completely merged with the edges of the original image, it is impractical to even re–engineer the index map by manual efforts.

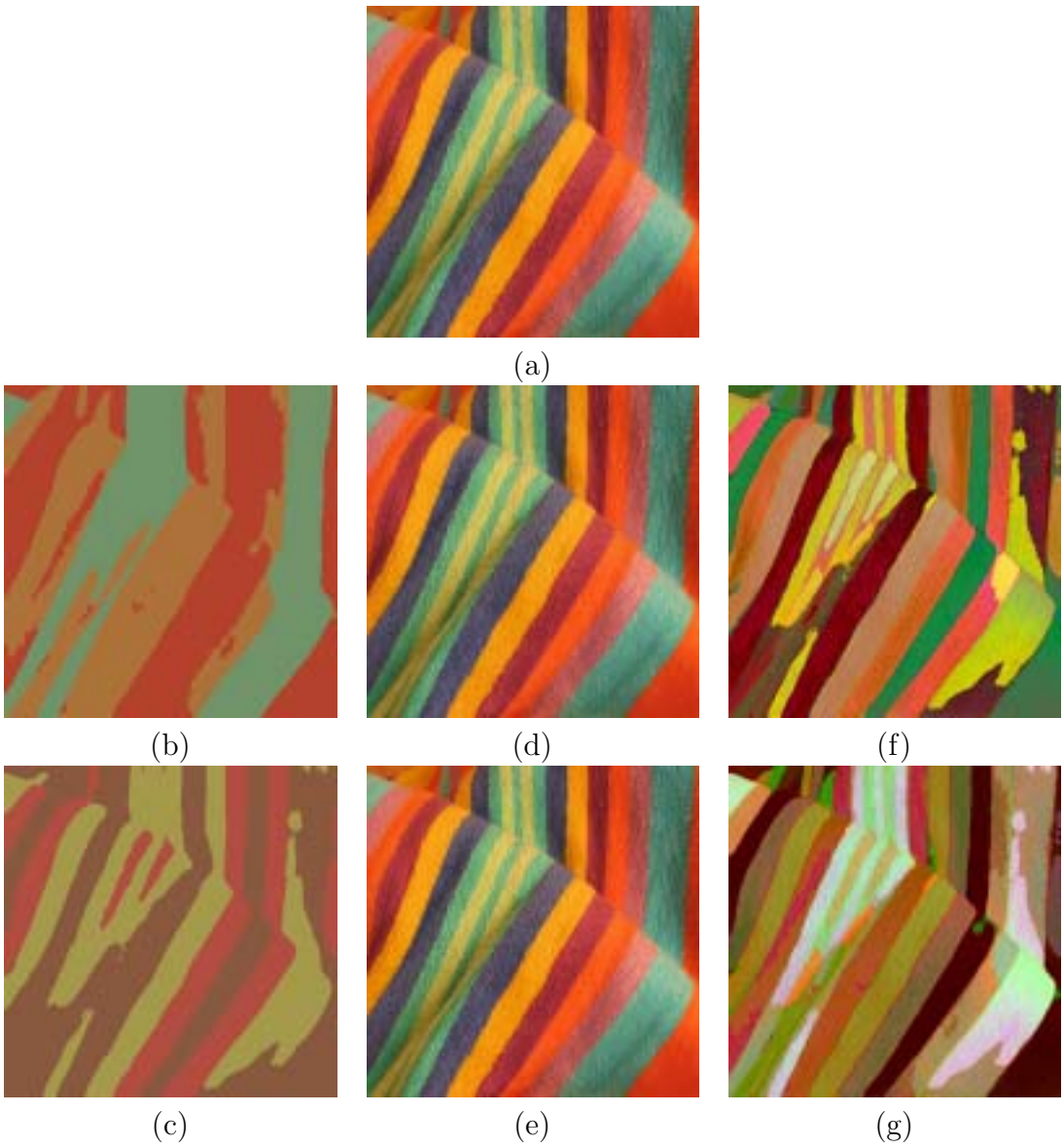


Figure 4.94: Decryption an image with an invalid key when C and m are known. (a) Original image. (b) and (c) Results of FPCAC. (d) and (e) Decryption with correct keys. (f) and (g) Decryption with invalid (switched) keys.

4.9.2 Block–Based Approach

The whole information used for encrypting I into I_e includes N, A, q, scale, bias, and $3q2^{2N}$ bits for the distortion angles. While each of A and q are represented using a nibble, writing A as $2^{-a} \times 2\pi$, $a \in \{0, \cdots, 7\}$ also lets A to be stored in a nibble. Hence, the total size of the encryption key equals $3q2^{2N} + 32$ bits (2 bytes are used for string the minimum and maximum of I_e to reconstruct the scale and bias). As for a known q there are q choices for each angel, there are $2^{3q2^{2N}}$ ways to encrypt a single image. In another view, there is a $2^{-3q2^{2N}}$ chance to crack the key of an image using a single trial. For a nominal value of $q = 3$ and $N = 3$, the total size of the key equals 76 bytes with 2.4×10^{173} possible solutions and 4.2×10^{-172} chance for cracking the code by a single trial. While, the chance to crack the code for a single block is 2^{-3q} (In this nominal case, 0.002), we emphasize that independent decryption of different blocks results in visible grids in the image. Hence, we argue that it is definitely impossible to crack the code by trial–and–error, even in these small choices for q and N.

Now, lets compte the computational complexity of the proposed method. In the encryption stage, each block is first separately processed. In this way, the processor should spent $3WH2^{-2N}$ flops on processing the expectation vector of a block and then 3 flops to compute the three angles. After spending 87 flops for computing V_{ij}, $15WH2^{-2N}$ flops are needed to perform the operation in (3.8.200). Adding up $3WH$ flops for computing the scale and bias and applying them, the total flops needed to

perform the proposed encryption process on an image equals $21WH + 105 \times 2^{2N}$. For nominal values of $W = H = 512$ and $N = 5$, 105×2^{2N} is less than two percents of $21WH$, hence, the total computation cost of the proposed encryption process equals $21WH$ flops, independent of N. The load for decrypting an encrypted image is equal to the above figure, except for $2WH$ flops spent on finding the bias and scale. Hence, decryption process elapses $19WH$ flops. In this way, the computational complexity of both process are linearly relating to the image size. To have an idea about the elapsed time of the proposed process on a DSP processor, we assume implementation of the method on a one Giga flop per second processor like $DIOPSIS^{TM}740$ by *Atmel* [12] or $TMS320C6713$ by *Texas Instruments* [13]. Encrypting a 512×512 image in these platforms elapses less than $6ms$ ($5ms$ for decryption). Hence, real–time video encryption using the proposed method only elapses 15% of the processor's time. Note that here the interesting features of this processors for pipelining and SIMD operation are ignored.

Figure 4.95 shows a few images and their corresponding encrypted versions. The parameters in this experiment are set as $N = 3$, $A = \frac{\pi}{2}$, and $q = 4$. While the key length equals 100 bytes (0.0127% of the image size), the chance for cracking the code by a single trial is less than 10^{-231}. In this case, t_e is less than $400ms$ and t_d is about $320ms$. Note that, while a single encryption or decryption stage lasts less than half a second, the PSNR of the reconstructed image is $44dB$, fairly higher than the

$38dB$ threshold indicated for professional satisfaction [207]. the maximum deviation is $34dB$ in all cases.

Figure 4.96, 4.97, and 4.98 illustrate the results of using different values of N, A, and q. According to the complete visual similarity of the original images and the decrypted ones, we only visualize the original images in the following parts of this thesis.

In Figure 4.96, in all cases $A = \frac{\pi}{2}$ and $q = 4$ are used. The elapsed time and final quality are almost the same for all the images. Both the encryption and the decryption stages elapse almost equally $400ms$ in all cases and the PSNR and maximum deviation values are $44dB$ and $34dB$, respectively. The key size varies from 5.5 bytes to 6148 bytes, resulting in a redundancy of 0.000699% to 0.782%. The chance to crack the code by a single trial changes from 0.00024 to less than 10^{-1470}. Hence, selecting a smaller N results in a much smaller code (and lower redundancy), but declines the security of the code. Also, N has no effect on the elapsed time and the quality of the reconstructed image. From this experiment and others we select $N = 3$ as a proper choice.

In Figure 4.97, $N = 3$ and $q = 4$ with different values of A are used. As expected, elapsed time has no dependency on A ($t_e \simeq 420ms$ and $t_d = 340ms$). Though, the PSNR and maximum deviation depend on A. as A increases from 2π to $\frac{1}{32}\pi$, PSNR increases from $41dB$ to $47dB$ and maximum deviation increases from $26dB$

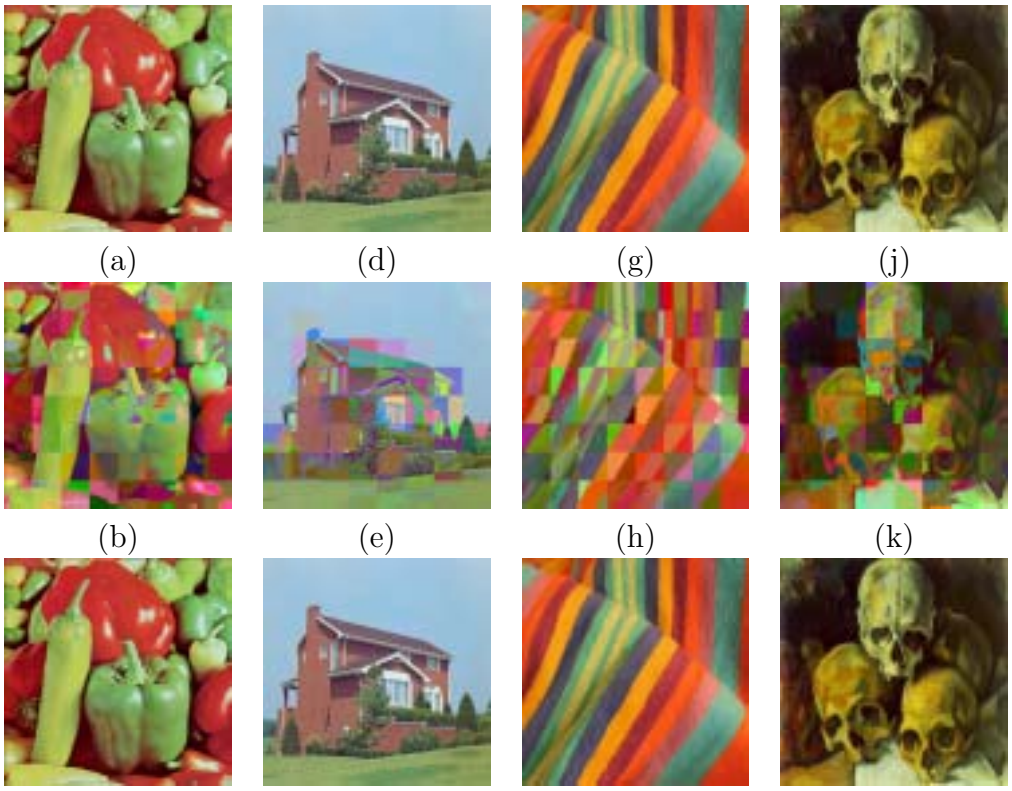


Figure 4.95: Sample results of the proposed encryption/decryption method. (a), (d), (g), and (j) Original images. (b), (e), (h), and (k) Encrypted images. (c), (f), (i), and (l) Decrypted images. (a) *Peppers*, courtesy of Signal and Image Processing Institute at the University of Southern California. (d) *House*, courtesy of The Ohio State University. (g) *Skulls*, adopted from *www.artchive.com* .

Figure 4.96: Effect of the choice of N. Image courtesy of *Shohreh Tabatabaii Seifi* and *Ali Qanavati*.

to $42dB$. This happens because smaller A results in less manipulations of data, resulting in less occurrences of numerical errors. As A lessens, the amount of visual distortions declines. In fact, when $A = \frac{1}{32}\pi$, there are literally no artifacts present in the encrypted image. As a compromise we use $A = \frac{1}{2}\pi$ in the rest.

In Figure 4.98, $N = 3$ and $A = \frac{1}{2}\pi$ with different values of q are used. As expected, the elapsed time does not depend on q ($t_e = 410ms$ and $t_d = 310ms$). Also, PSNR and maximum deviation are almost still (PSNR equals $43dB$ and maximum deviation equals $33dB$). But the key size varies from 4 bytes to 172 bytes and the probability of cracking the code by a single trial decreases from 1 to 3×10^{-410}. As a compromise we select $q = 4$.

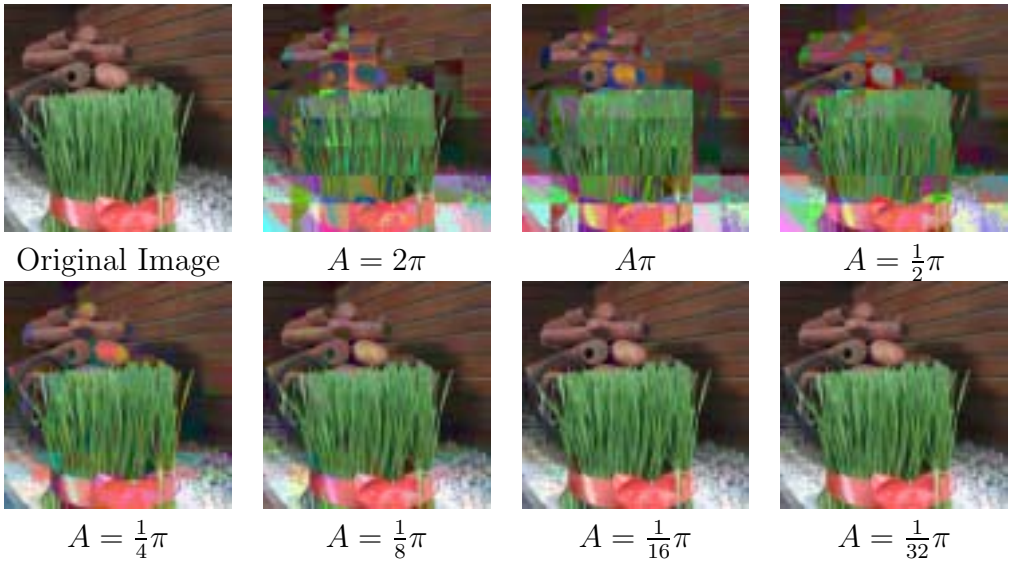


Figure 4.97: Effect of the choice of A. Image courtesy of *Shohreh Tabatabaii Seifi* and *Ali Qanavati*.

Figure 4.99 illustrates a trial to decrypt an image using a key from a different encryption code of its own. In this way, Figure 4.99–a shows an image encrypted to the images shown in Figures 4.99–b and c. Now, decrypting the image with its own key results the images shown in Figures 4.99–d and e, while switching the keys results in the images shown in Figures 4.99–f and g. While an image may be encrypted in $2^{3q2^{2N}}$ ways, no two encryptions share a similar key.

Figure 4.100 investigates the possibility of reproducing I_d from the compressed version of I_e. Figure 4.100–a shows a part of the original image and Figure 4.100–b and Figure 4.100–d illustrate the corresponding section of the results of the proposed decryption method when the encrypted image is, correctly, compressed using JPEG2000

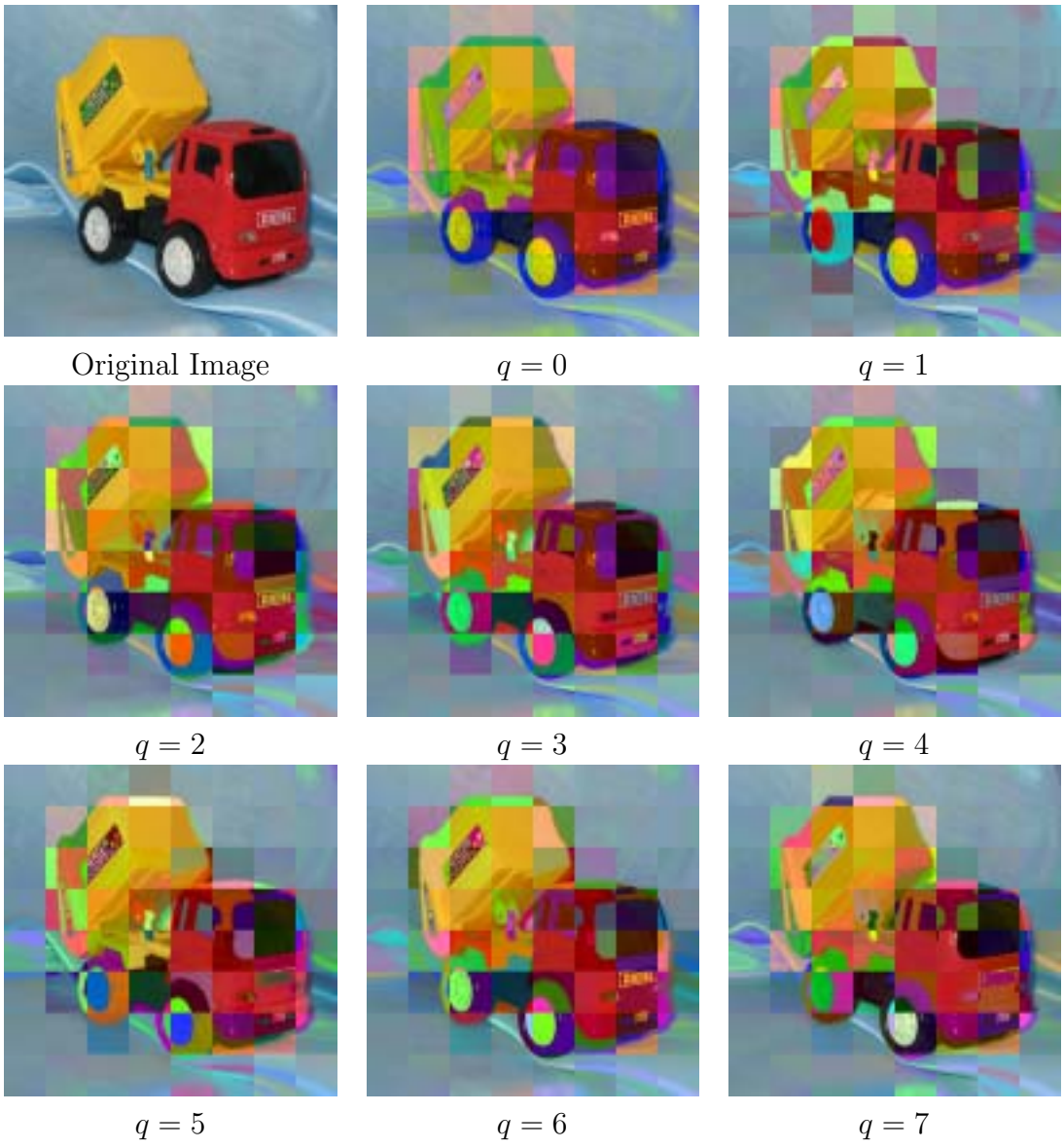


Figure 4.98: Effect of the choice of q.

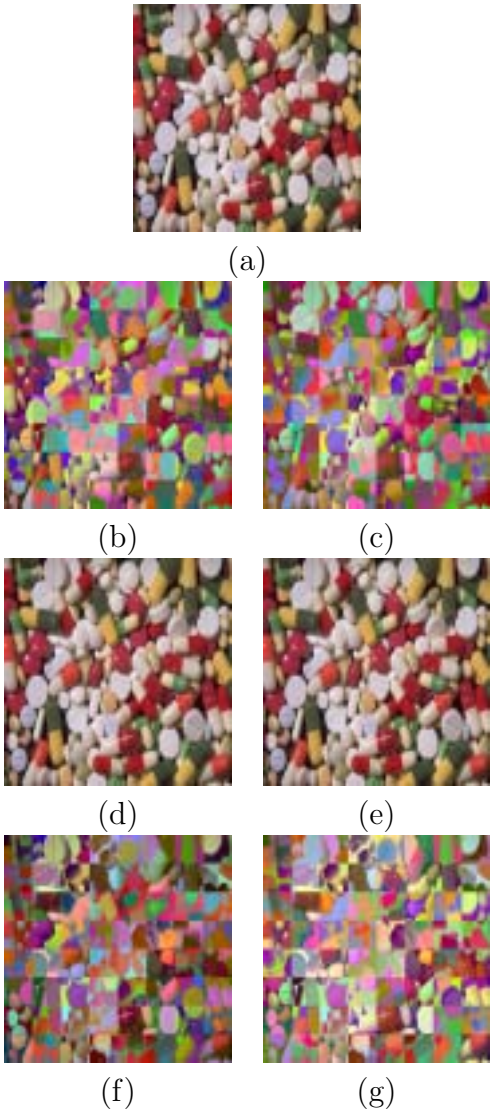


Figure 4.99: Trial to decrypt an image using a key from a different encryption code of its own. (a) Original image. *Pills*, courtesy of *Karel de Gendre*. (b) and (c) Different encryptions. (d) and (e) Decryption using own keys. (f) and (g) Decryption using each other's keys.

and JPEG (without chrominance down–sampling). Here, JPEG2000 compression is utilized using *Advanced Batch Converter* 3.9 and JPEG compression is performed using *ACDSee* 5.0. In this way, JPEG2000 is utilized with rate of 90 and JPEG is performed with quality of 70. Figure 4.100–b shows the results of using the JPEG compression module incorporated into MATLAB, which performs the chrominance down–sampling. In this case, the quality of compression is selected 100. The fake lines are present when JPEG with chrominance down–sampling is used.

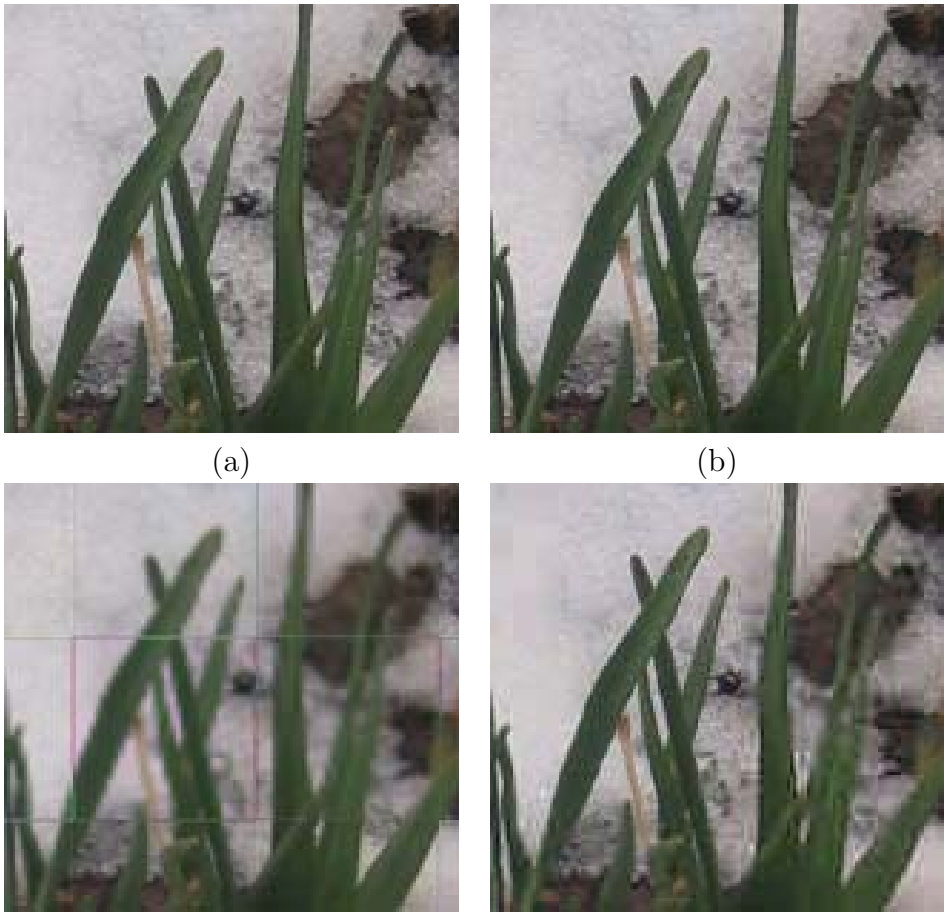
Figure 4.101 shows a sample utilization of the proposed method in the image sequence *Clair*. Each process of encryption or decryption of a single frame elapsed less than $100ms$ in this experiment and the PSNR is $44dB$ with maximum deviation of $34dB$.

4.10 Color Transfer

4.10.1 Colorizing

Figure 4.102 shows the segmented version of the three standard images of *Lena*, *Girl*, and *Barbara* along with some samples adopted from *Welsh et al.* [375] and *Levin et al.* [238]. The segmentation is performed manually with repeatedly use the magic select tool in *Adobe Photoshop* into six distinct segments like the skin, hair, lip, cloth, and the force.

Figures 4.103 and 4.104 show the results of applying the two proposed colorizing methods of (3.9.206) and (3.9.207) on the two samples of *Welsh et al.* [375] along



(a) (b)

(c) (d)

Figure 4.100: Adaptivity of JPEG–based compression and the proposed method. a) Original image. b) JPEG2000. c) JPEG with chrominance down–sampling. d) JPEG without chrominance down–sampling.

Figure 4.101: Utilization of the proposed method in the image sequence *Clair*.

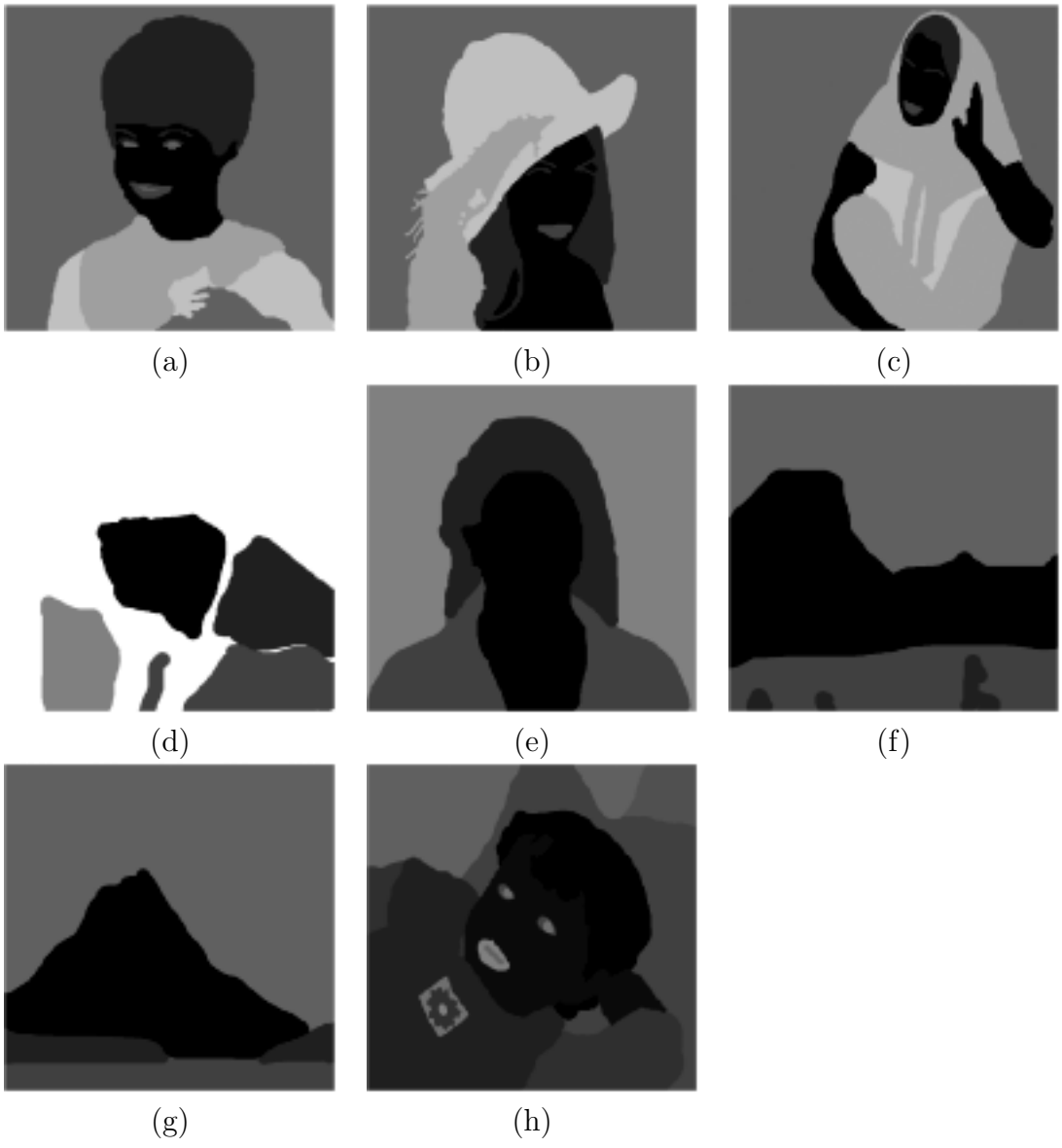
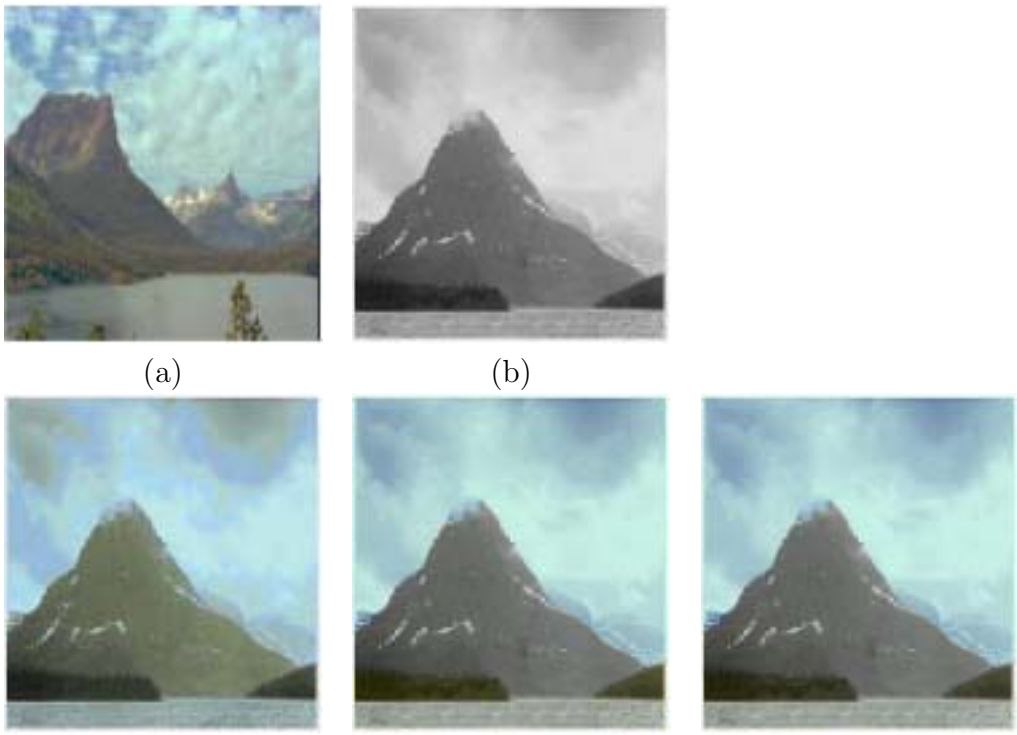


Figure 4.102: Segmented version of test images, (a) *Girl*, (b) *Lena*, (c) *Barbara*, (d),(e),(f),(g) adopted from *Welsh et al.* [375], and (h) adopted from *Levin et al.* [238].



(a) (b)

(c) (d) (e)

Figure 4.103: A sample from [375] along with the results of the proposed methods. (a) Reference image, (b) Source image. (c) Result of [375]. (d) Result of (3.9.206). (d) Results of (3.9.207). (a),(b),(c) Adopted from [375].

with the results of their method.

The subjective test of goodness on *Welsh et al.* [375] samples gave the following results. When the average score for *Welsh et al.*'s algorithm [375] is 1.833, the method in (3.9.206) is scored 2.119 and (3.9.207) is given 2.429 making the average score for the methods about 2.27 (all out of 3). Thus, The methods are recognized subjectively, 16% and 32% better than *Welsh et al.*'s results [375], respectively. This results in 24% better scores in average. Also, the second proposed method is about 15% better

(a) (b)

(c) (d) (e)

Figure 4.104: A sample from [375] along with the results of the proposed methods. (a) Reference image, (b) Source image. (c) Result of [375]. (d) Result of (3.9.206). (d) Results of (3.9.207). (a),(b),(c) Adopted from [375].

than the first method.

Welsh et al. [375] use the luminance as the segmentation criterion. Hence, their method is completely dependence on the similarity of the source and the reference images in the grayscale domain. Using the intensity as the matching feature makes an odd perceptual artifact in *Welsh et al.*'s [375] synthesized images as a discontinuity in color appearance. Such effect can be easily recognized in the background of the samples shown in Figure 4.103–c and Figure 4.104–c and other samples shown in [375]. Also, the methods perform PCA for each sample; in contrast with *Welsh et al.*'s [375] work that uses the $\ell\alpha\beta$ color space that is based on performing the PCA on a set of sample images to derive the color space formulation. Therefore, (theoretically) we are working on a better base of uncorrelatedness.

The three conditions defined in Section 3.9.1 are a ground–truth for new colorizing algorithms. The first proposed method, justifies only two of the conditions, where the second proposed method justifies all. *Welsh et al.*'s [375] method holds the first condition, as he just transfers α and β components, leaving the ℓ value unchanged but they have given no attention to the two other conditions. It must be emphasized that the ℓ component of $\ell\alpha\beta$ is not a linear norm function.

The method works about 60 times faster than *Welsh et al.*'s method [375]. The proposed methods averagely elapses less than 2 seconds whilst *Welsh* [375] has reported a 15 seconds to 4 minutes record in similar platforms, though they offer to

Table 4.18: Comparison of *Welsh et al.*'s colorizing method [375] and the two proposed methods.

Method	**Goodness** $(1\cdots3)$	**Criterion**			**Elapsed Time**
		I	**II**	**III**	
Welsh et al. [375]	1.83	$\surd$	–	–	$15s$–$4min$
Proposed (3.9.206)	2.12	$\surd$	–	$\surd$	$2s$
Proposed (3.9.207)	2.43	$\surd$	$\surd$	$\surd$	$2s$

use larger neighborhood windows to enhance the results. Table 4.18 concludes the comparison between *Welsh et al.*'s work [375] and the proposed algorithms.

The grayscale version of the three face images, *Lena*, *Girl*, and *Barbara* (see Figure 4.105), are colorized 12 times, once using *Lena* as the reference image and then using *Girl* for this purpose, each colorization performed twice, once by (3.9.206) and then by (3.9.207). A subjective test is performed on the results, containing two questions of impairment (Table 3.2) and *goodness* (Table 3.3). As the background is retained grayscale in all synthetic images, the background of the original images were made grayscale before exposure to the test subjects. To find out the method's performance when dealing with the special problem of face colorizing, both questions are asked twice, once for the entire image and then only for the face area. As the color version of *Barbara* is not available, only the goodness question is asked regarding this image. To measure the quality of the proposed colorizing schemes, sample color images are converted to grayscale and then colorized by their own color information or others'. In this way PSNR values of colorizing *Lena* and *Girl* (with each other)

Figure 4.105: Results of the proposed colorizing methods. (a),(b),(e),(f),(i), and (j) Reference image is *Lena*. (c),(d),(g),(h),(k), and (l) Reference image is *Girl*. (a),(e),(i),(c),(g), and (k) Colorized by (3.9.206). (b),(d),(f),(h),(j), and (l) Colorized by (3.9.207).

in both proposed methods are listed in Table 4.19. For a 256×256 image containing eight regions, the color extraction phase is performed in 2 seconds averagely, when the color transfer phase got another 2 seconds. Results of subjective test on face colorizing scaled to the range of $[0 \ldots 100]$ are listed in Table 4.20.

Table 4.19 shows that both proposed methods are similar in the PSNR sense.

Table 4.19: PSNR values for *Lena* and *Girl* colorizing tests, using different methods.

Reference		*Lena*		*Girl*	
Method		(3.9.206)	(3.9.207)	(3.9.206)	(3.9.207)
Source	*Lena*	$26dB$	$26dB$	$18dB$	$21dB$
	Girl	$14dB$	$13dB$	$28dB$	$28dB$

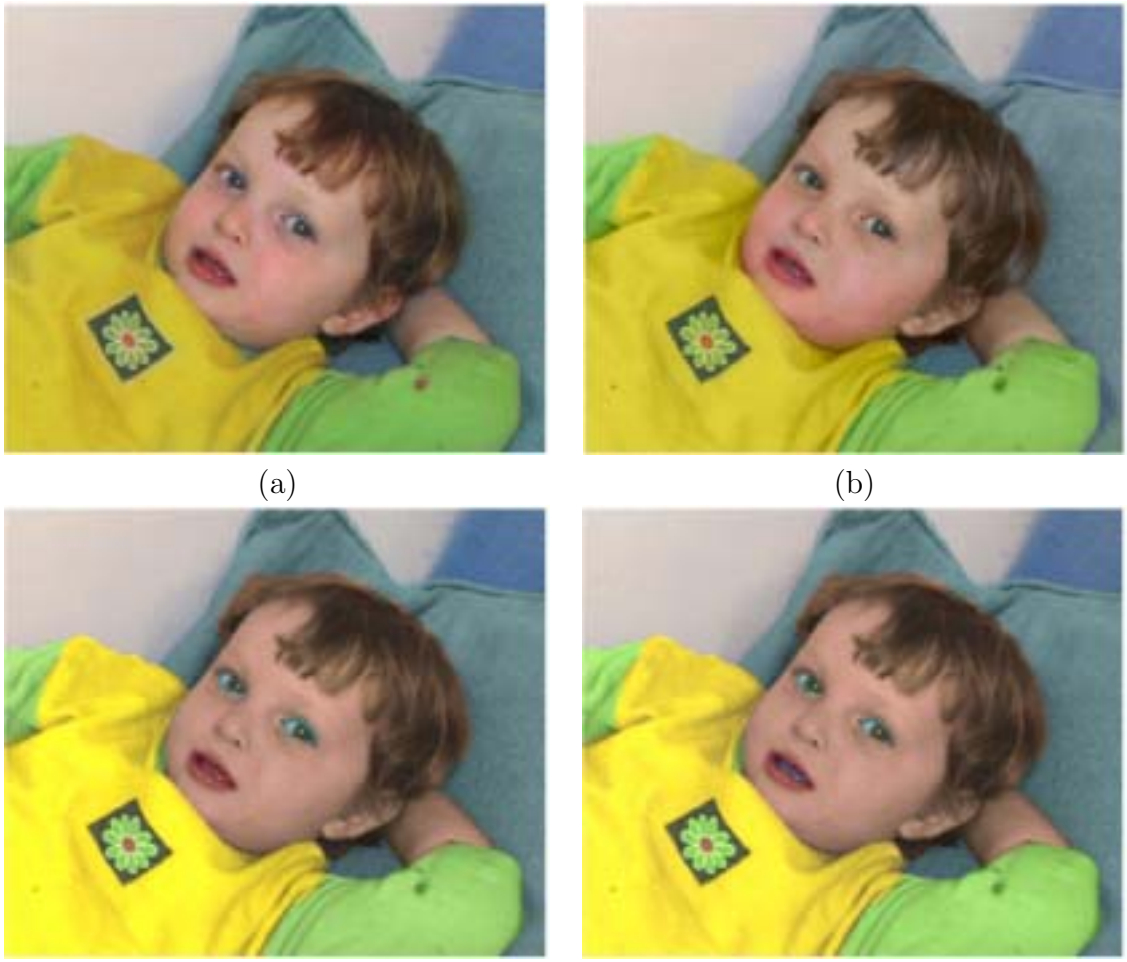
Table 4.20: Subjective tests of the colorizing tests, all values are out of 100.

Condition	**Method**	**Image**		**Face**	
		Impairment	**Goodness**	**Impairment**	**Goodness**
$M = S$ [a]	(3.9.206)	67	75	82	90
	(3.9.207)	75	87	88	98
$M \neq S$ [b]	(3.9.206)	35	72	45	80
	(3.9.207)	40	83	52	96
NoS [c]	(3.9.206)	–	60	–	80
	(3.9.207)	–	78	–	95

[a] Reference image is the colored version of the source image.
[b] Another image is used as the reference image.
[c] Colored version of the source image is not available.

Colorizing each image by its color information gives better results, though the meaning of the PSNR test must be considered carefully. Many objectionable images with high values of PSNR can be produced. In face images, the proposed methods are approximately scored 83 out of 100, which means the synthetic images are averagely perceived "*good*". In impairment scales, the colorized images are considered "*not objectionable*" (60.5 out of 100). Face region is averagely ranked 23% higher than the whole colorized area and the method proposed in (3.9.207) is ranked 18% better than (3.9.206). Also, it must be emphasized that colorizing an image with its own color content, gives better results, but not much better (less than 6%). In the PSNR sense colorizing a color image using its own color information is massively outperforming the case of colorizing a color image using a different reference image.

Figure 4.106 shows the comparison of the proposed colorizing method with the method by *Levin et al.* [238]. Their method elapses 15 seconds on each image compared to less than 2 seconds record of the proposed methods. Comparing the visual appearance of the results, their method is performing better in the edges because of the interpolation scheme they are using. We propose to use their novel optimization framework combined with the PCA–based color generation stage to reach better results. In terms of the defined criteria, the method by *Levin et al.* [238] fulfills the first condition (not altering the original grayvalue) but fails the two others.



(a) (b)

(c) (d)

Figure 4.106: Comparison of the proposed colorizing methods with the method by *Levin et al.* [238]. (a) Original image adopted from *Levin et al.* [238]. (b) Result of [238]. (c) Result of the method proposed in (3.9.206). (d) Result of the method proposed in (3.9.207).

4.10.2 Query Region–Based Recoloring

Figure 4.107 shows some of the sample images used for evaluating the performance of the proposed color transfer method. The proposed color transform method is performed on some typical sample images shown in Figure 4.107. The results of the proposed method along with results of other researcher's methods are shown in Figure 4.108. The β values and the description of the selected regions in images are expressed in the caption of Figure 4.108. It is worth to note that transferring color information of a 512×512 image into a 512×512 image according to four selected medium–sized base swatches elapses less than 2 seconds.

Comparing the results shown in Figure 4.108 and Figure 4.109 shows the difference among various algorithms. Figure 4.109–a, 4.109–b, and 4.109–f show sample results of *Reinhard et al.* [297]. Comparing Figure 4.109–a with Figure 4.108–a shows that the results of the proposed method and that method are almost similar. In Figure 4.109–b the differences are clear. *Reinhard et al.* [297]'s method have changed the color of the whole image to the same greenish sense, resulting in blurring the strong edges between the trees and the sky. Also, the blue sky in Figure 4.109–b has nothing to do with the entirely white sky of Figure 4.107–f. In addition, the method in [297] has eliminated the red flowers. In contrast, the result of the proposed method in Figure 4.108–b shows no blurring and proper color contrast. Also the warm colors of the red flowers are completely reserved. Although, *Greenfield*'s method [156] has tried to transfer

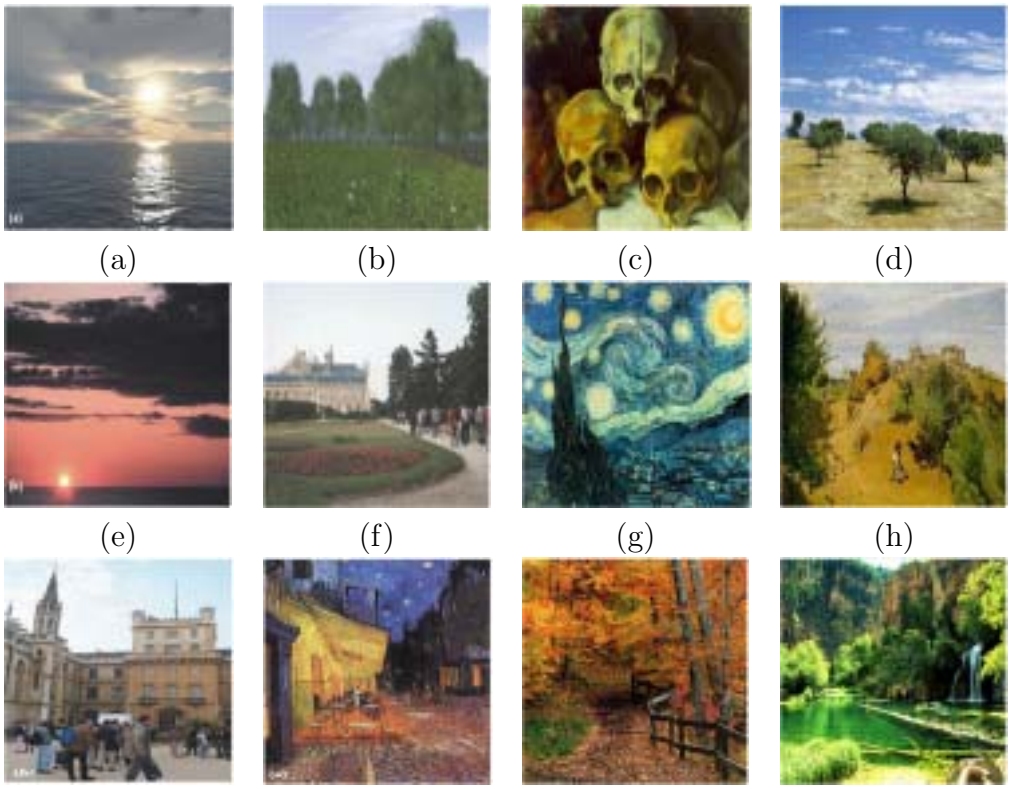
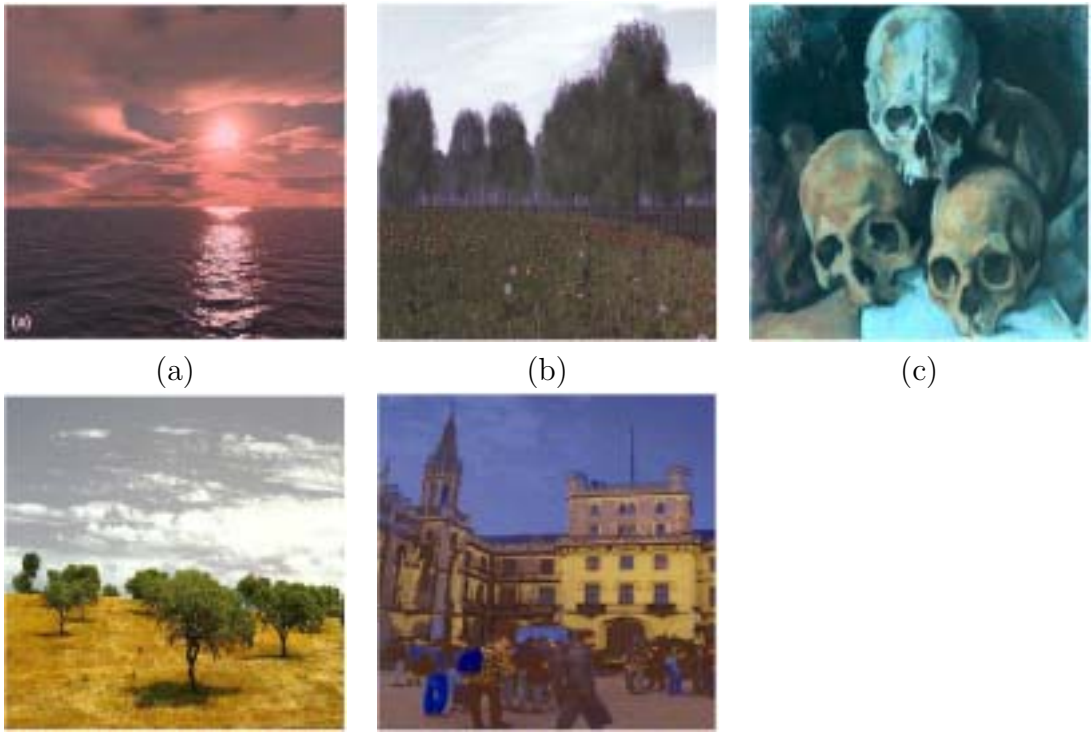


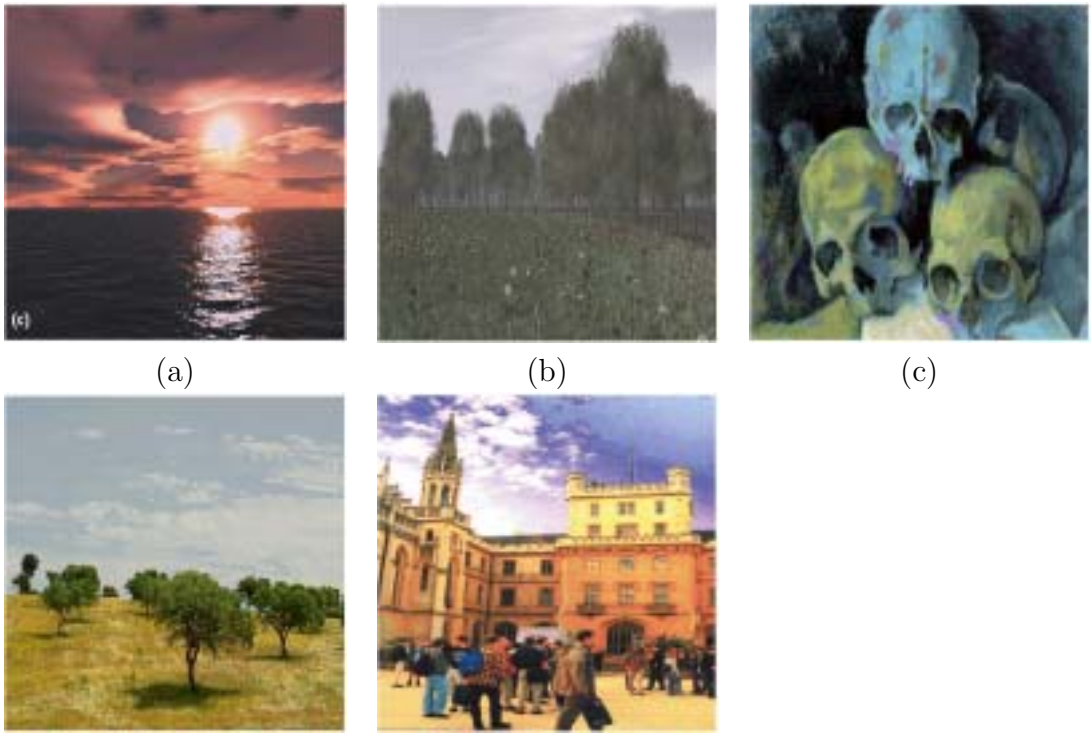
Figure 4.107: Sample images. (a), (b), (c), (d), (i), and (k) Source images. (e), (f), (g), (h), (j), and (l) Reference images. (a), (b), (e), (f), (i) and (j) Adopted from *Reinhard et. a.* [297]. (c), (g) Adopted from *Greenfield et al.* [156]. (d), (h) Adopted from *Chang et al.* [92]. (k), (l) Adopted from *www.webshots.com* with permission of the authors. (k) *Mc. Cormic Creck State Park, Indiana* by *Mike Briner*, *mbphoto@spraynet.com*, *www.mikebrinerphoto.com*. (l) *Hanging Lake* by *Brent Reed*, *brent@reedservices.com*.



(a) (b) (c)

(d) (e)

Figure 4.108: Results of the proposed recoloring method applied on sample images of Figure 4.107. Source images are shown in Figures 4.107–a, 4.107–b, 4.107–c, 4.107–d, 4.107–i, and 4.107–k, respectively. Reference images are shown in Figures 4.107–e, 4.107–f, 4.107–g, 4.107–h, 4.107–j, and 4.107–k respectively. Used settings are (a) $\beta = 0.95$, $R = \{Sky, Water\}$, (b) $\beta = 0.1$, $R = \{Leaves, Sky\}$, (c) $\beta = 0.8$, $R = \{Skulls, Background\}$, (d) $\beta = 0.5$, $R = \{Leaves, Sky, Earth\}$, (e) $\beta = 0.95$, $R = \{Sky, Building, Pavement\}$.



(a) (b) (c)

(d) (e)

Figure 4.109: Results of other available methods applied on sample images of Figure 4.107. Source images are shown in Figures 4.107–a, 4.107–b, 4.107–c, 4.107–d, and 4.107–i, respectively. Reference images are shown in Figure 4.107–e, 4.107–f, 4.107–g, 4.107–h, and 4.107–j respectively. (a),(b),(e) Method in [297]. (c) Method in [156]. (d) Method in [92].

the cold colors of Figure 4.107–g to Figure 4.107–c, Figure 4.109–c still contains the warm violet–blue colors on the topmost skull. In the result of the proposed method in Figure 4.108–e, the colors are mostly cold. Figure 4.109–d shows the result of the method by *Chang et al.* [92] which is almost acceptable but lacks proper color contrast. Returning back to *Reinhard et al.*'s method [297], there is a discontinuity artifact in the sky in Figure 4.109–e. Note that, the sky in Figure 4.108–e simulates a real night sky having in mind that the photograph has taken at daytime. This fact is more desiring when one confirms that the recolorized image by the method in [297] shown in Figure 4.109–e does not seem to be a night scene while Figure 4.108–e does seem so.

Although, transferring the color information of a real image to a painting and vice versa is possible, one must consider the meaning of such transformation thoughtfully. While the color information in a real photograph is a continually changing vector, it is a regional almost constant information in a painting (due to the process that has made it; the brush with a unique color moves on the canvas). Furthermore, using a reference image of entirely different scene compared with the source image does not make the process to fail, but the results of such operation must be deliberated carefully. The same event occurs when giving regions of the source image and the reference image in a scattered fashion; for example trying to transfer color information of leaves to sand. The exact report of the time measurement is neglected in the references but

considering the less than two seconds record of the proposed method while other methods use sophisticated methods of segmentation and convex hull computation, the outperforming results of the proposed method is clear. It must be emphasized that in the proposed methods, using an image as source image and reference image at the same time (recoloring an image with itself as the reference image), when working on almost the same regions in the reference image and the source image, results in an image that cannot be recognized from the original image. In addition when using single–swatch version of the method, the process is completely revertible. Although in all samples discussed above the reference image is unique, there is no limitation that prevents the user from using two or more images as the reference image. That is because the proposed method is not using the referencee image actually, but it needs a set of homogenous swatches. This option may be useful when trying to recolor a source image due to the sky in the first reference image and the leaves in the second reference image two. Such option is available in the method in [297] but not in the methods in [156, 92].

The proposed method needs a set of homogenous swatches in the two images as the reference for the transform and a single β parameter that controls the averaging of the color vectors. We wish to emphasize that this is the minimal set of givens for this problem. the methods in [92, 156] transfer the color information based on a set of fixed color categories or some spatial cues, leaving no room for user

intervention. What we really need from a proper color transfer methods is to let the user intentionally select a set of relations in the color domain and ask the method to perform the suitable transformation. Such input to the algorithm can be eliminated if we reach upon intelligent algorithms that understand the image contents to find the relation among the scene materials automatically. We are insisting that it is not the shortage of an algorithm but it is its user friendliness to make him able to give his intention. The β parameter is the one neglected in available recoloring methods. It professionally controls the blurring attitude of the method. Due to different imaging conditions, it is probable that the reference and the source images are in different conditions of edges. Then, by using β, one can compensate these differences.

To make the power of the proposed color transform process more clear, the source image of Figure 4.107–k and the reference image of Figure 4.107–l are shown in Figure 4.110 again, along with the recolorized version.

4.10.3 Clustering based Recoloring

Figure 4.111 shows a sample run of the proposed method containing the original images along with the intermediate and final results. Here, each of the images shown in Figure 4.111–a and Figure 4.111–b are recolored using each other as the reference image. Figure 4.111–c and Figure 4.111–d show the corresponding segmentation results given by the FPCAC and Figure 4.111–e and Figure 4.111–f illustrate the corresponding pallets. When recoloring Figures 4.111–a according to Figures 4.111–b

(a)

(b) (c)

Figure 4.110: Results of the proposed colorizing method. (a) The reference image. (b) The source image. (c) The result of the proposed recoloring method ($\beta = 0.2$, $R = \{Leaves, Bushes, Bark\}$).

Figure 4.111: Results of the proposed recoloring method. (a) and (b) Original images. (c) Segmentation results of a by the FPCAC. (d) Segmentation of b by the FPCAC. (e) The corresponding pallets (Top: a, Bottom: b). (f) The destination when a is the source and b is the reference. (g) The destination when b is the source and a is the reference.

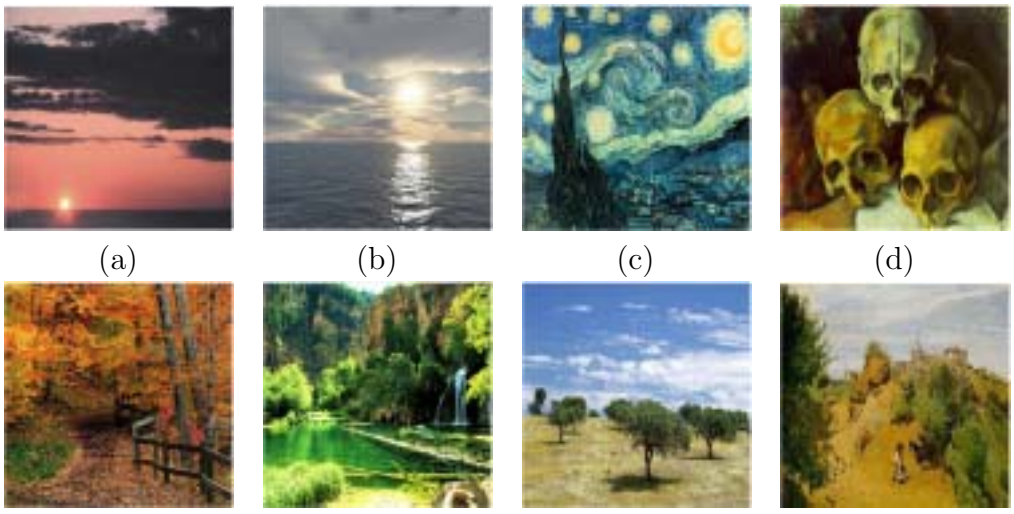


Figure 4.112: Sample images. (a) and (b) Adopted from [297]. (c) and (d) Adopted from [156]. (e) and (f) Adopted from *www.webshots.com* with permission of the authors. (e) *Mc. Cormic Creck State Park, Indiana* by *Mike Briner*, *mbphoto@spraynet.com*, *www.mikebrinerphoto.com*. (f) *Hanging Lake* by *Brent Reed*, *brent@reedservices.com*. (g) and (h) Adopted from [92].

the pallet assignments is as 311 meaning that the first pallet representative in the source image is assigned to the third one in the reference image, while the two others are assigned to the first one. In the reverse transform, the assignment is also 311. Both operation take about 11 seconds.

Figure 4.112 shows other sample images used in this thesis. The proposed recoloring method is performed on all images both as the source image and the reference image. Figure 4.113 shows the results. Figure 4.114 shows a few results of other approaches which were available to the author. Table 4.21 lists the corresponding source and reference images along with the elapsed times. The exact report of the

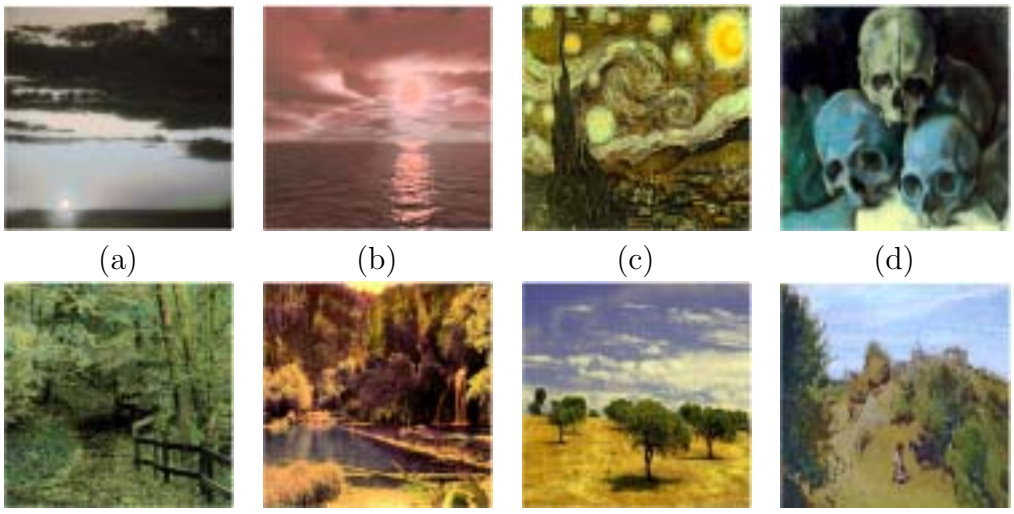
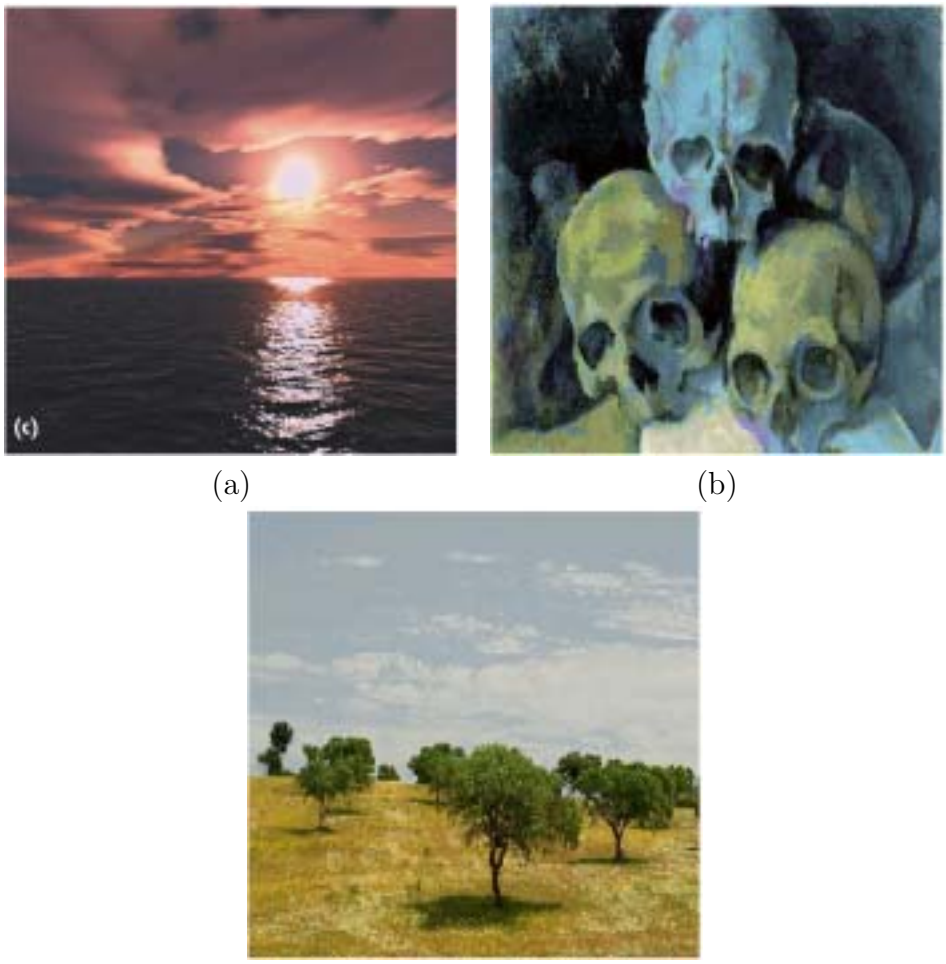


Figure 4.113: Results of the proposed method when working on the images shown in Figure 4.112. For details see Table 4.21.

time measurement is neglected in the references but considering the less than 30 seconds record of the proposed method while other methods use sophisticated methods of hierarchical segmentation and convex hull computation, the outperforming state of the proposed method is clear.

4.11 Image Registration

Figure 4.115 shows some different transformations of an image and Figure 4.116 shows the control points in the image shown in Figure 4.115–a. Figure 4.117 shows two panchromatic images taken from the city of *Bam* by the *IRS–1D* satellite PAN sensor, before and after the devastating earthquake of December 26, 2003. Figure 4.118 shows



(a) (b)

(c)

Figure 4.114: Results of other approaches. (a) *Reinhard*, *Ashikhmin*, *Gooch*, and *Shirley* [297]. (b) *Greenfield* and *House* [156]. (c) *Chang*, *Saito* , *Nakajima* [92]. For details see Table 4.21.

Table 4.21: Description of the results shown in Figure 4.113 and Figure 4.114.

Source	**Reference**	**Destination**	**Elapsed Time (s)**
4.112–a	4.112–b	4.113–a	16
4.112–b	4.112–a	4.113–b	16
4.112–c	4.112–d	4.113–c	8
4.112–d	4.112–c	4.113–d	8
4.112–e	4.112–f	4.113–e	8
4.112–f	4.112–e	4.113–f	8
4.112–g	4.112–h	4.113–g	6
4.112–h	4.112–g	4.113–h	6
4.112–b	4.112–a	4.114–a	–
4.112–d	4.112–c	4.114–b	–
4.112–g	4.112–h	4.114–c	–

the urban portion of the images. The first images are cropped with no magnification to focus on details.

4.11.1 Geometrical Estimation

Figure 4.119 shows the results of registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed geometric method. Here, the 20 control points shown in Figure 4.116 are used. Table 4.22 lists the values of total distortion and the elapsed time in this test. Table 4.23 lists the values of ε when registering the samples shown in Figure 4.115 on themselves and on each other. In this thesis, every where the nearest–neighborhood interpolation method is used.

Note that when the transformation is not massively changing the image (all except (e)), the proposed geometric registration method results in about 1% distortion. Also, note the low value of the standard deviation of distortion when registering a single

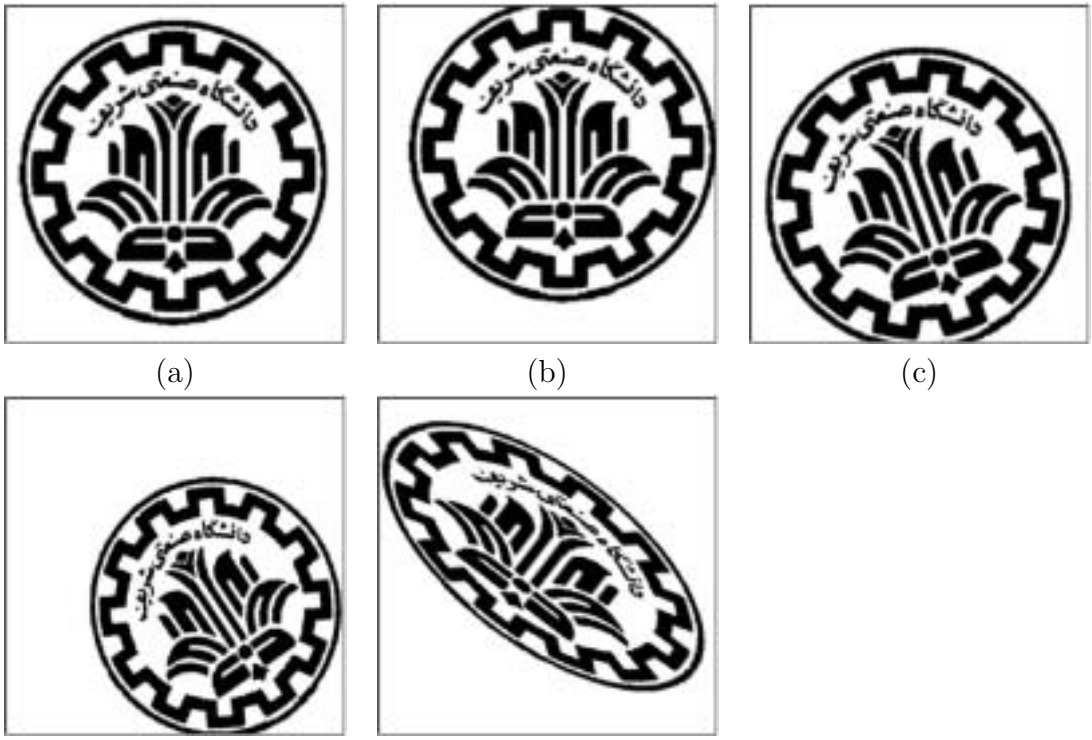
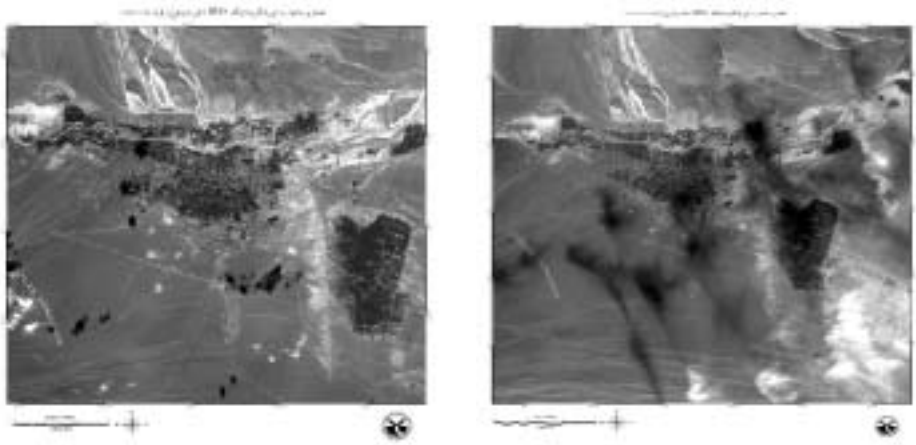


Figure 4.115: Different transformations of an image. (a) First image. (b) Translated. (c) Rotated and Translated. (d) Rotated, translated, and balanced scaled. (e) Rotated, translated, and unbalanced scaled.

Table 4.22: Elapsed time and total distortion results for registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed geometric method.

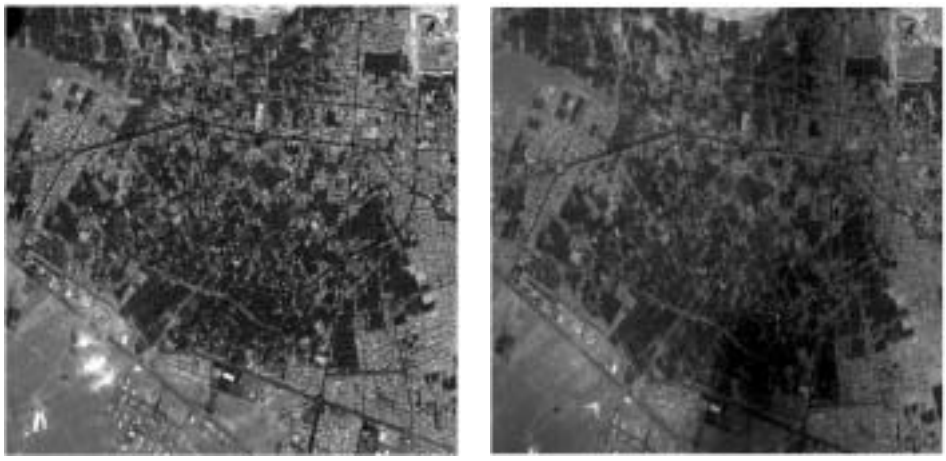
Second Image	(a)	(b)	(c)	(d)	(e)
t	$10ms$	$11ms$	$10ms$	$10ms$	$10ms$
ε	1.07%	1.07%	1.07%	1.06%s	1.07%

Figure 4.116: Control points on the image shown in Figure 4.115–a.



(a) (b)

Figure 4.117: Panchromatic images of the city of *Bam*. (a) 2003-12-04. (b) 2003-12-29.

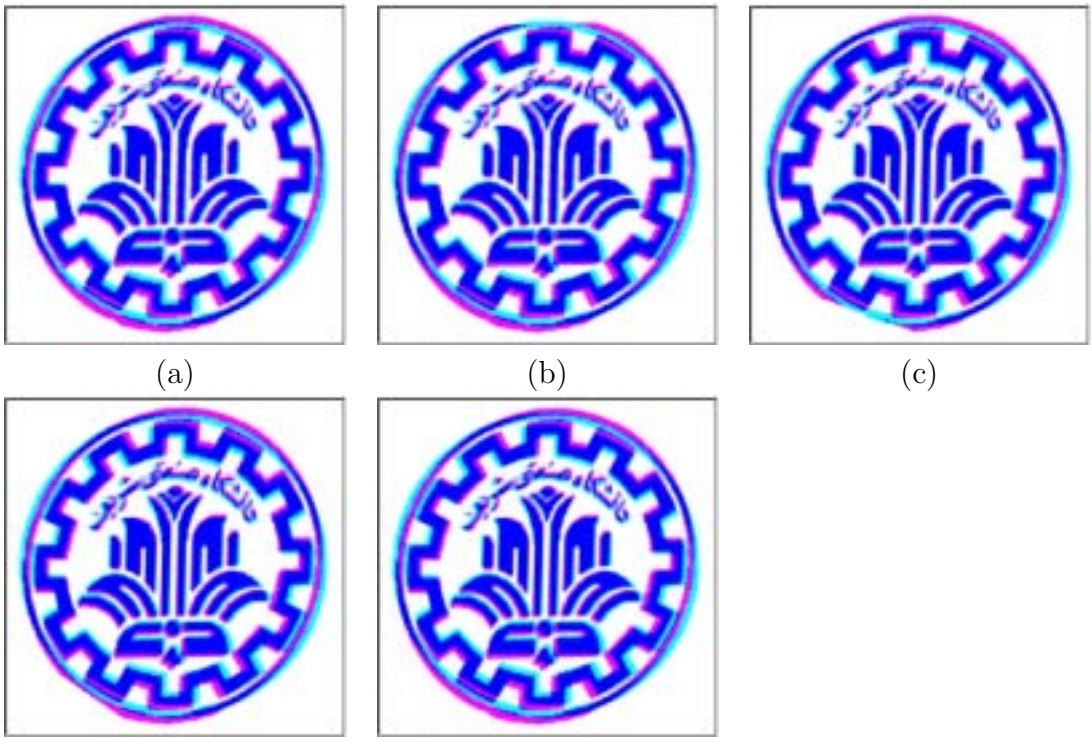


(a) (b)

Figure 4.118: Urban portion of the images shown in Figure 4.117.

Table 4.23: Total distortion results when registering the images shown in Figure 4.115 with each other and themselves using the proposed geometric method.

First Image / **Second Image**	(a)	(b)	(c)	(d)	(e)
(a)	1.07%	1.08%	1.07%	0.68%	4.35%
(b)	1.07%	1.07%	1.07%	0.67%	4.35%
(c)	1.08%	1.08%	1.07%	0.68%	4.35%
(d)	1.06%	1.07%	1.06%	0.66%	4.37%
(e)	1.07%	1.07%	1.07%	0.67%	4.35%
σ	0.0049%	0.0053%	0.0045%	0.0064%	0.0073%
η	1.0708%	1.0746%	1.0699%	0.6730%	4.3540%



(a) (b) (c)

(d) (e)

Figure 4.119: Results of applying the proposed geometric method when registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a.

Table 4.24: Elapsed time for registering images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed gradient–descent enhanced method.

Second Image	(a)	(b)	(c)	(d)	(e)
t	$5.2s$	$5.4s$	$0.1s$	$0.1s$	$2.8s$
ε	0.36%	0.29%	0.099%	0.099%s	0.1%

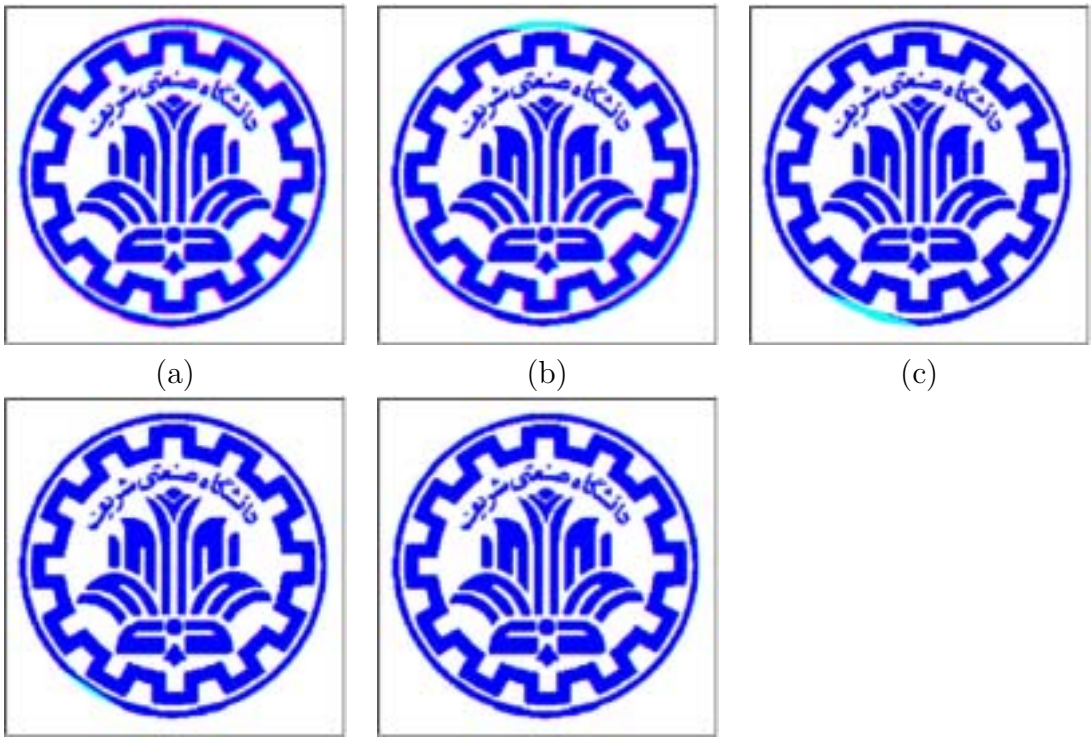
image on others, even when the distortion is about 4%. The average elapsed time in this test is $10ms$.

A test is performed to see wether double times applying the proposed geometric registration method on the second image enhances the results. In numerous images, it is observed that the value of ε and the resulting transformation are identical or negligibly different in the next application of the proposed method on the registered image.

4.11.2 Numerical Enhancement

Gradient–Descent Enhancement

Figure 4.120 shows the results of applying the proposed gradient–descent enhanced method with $\lambda = 0.002$ and $\tilde{\varepsilon} = 0.001$, (0.1% error). The maximum number of iterations is set as 10000. Table 4.24 lists the elapsed times and total distortion values. Note the better alignment of the images, compared with the results of the proposed geometric method shown in Figure 4.119, while the computational cost is much more.



(a) (b) (c)

(d) (e)

Figure 4.120: Results of using the gradient–descent enhanced proposed method for registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a.

Table 4.25: Elapsed time for registering images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed Newton enhanced method.

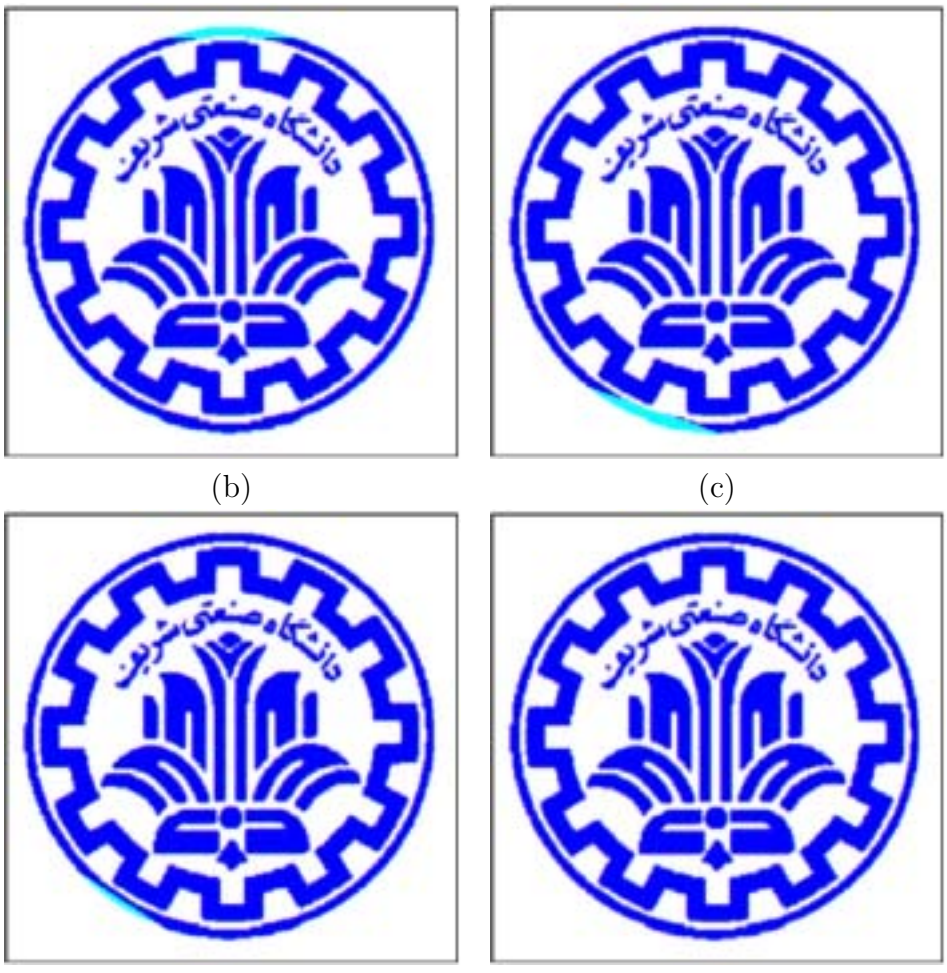
Second Image	(b)	(c)	(d)	(e)
t	$16ms$	$15ms$	$16ms$	$17ms$
ε	0.02%	0.05%	0.05%	$0.06ms$

Newton Enhancement

Figure 4.121 shows the results of applying the proposed Newton enhanced method with $\varepsilon_\circ = 10^{-10}$. Table 4.25 lists the elapsed times and total distortion values. Note the perfect alignment of the images, while the elapsed time is less than $20ms$ in all cases. The proposed Newton enhanced method vanishes for registering an image on itself, because the Hessian matrices get singular. Table 4.26 lists the values of total distortion resulted when registering the images shown in Figure 4.115 on each other using the proposed Newton enhanced method. Figures 4.122, 4.123 show the results of registering the images shown in Figures 4.117,4.118, respectively, using the proposed Newton enhanced method. The resulting total distortion values are $\varepsilon = 0.011\%$ and $\varepsilon = 0.566\%$, respectively, while both has elapsed about $15ms$. Note that the algorithm speed is independent of the image size, which is of great important in working with large remote sensing images.

4.11.3 Control Point Distribution and Distortions

In this section we investigate the effects of the distribution of the control points on compensating distortions. Figure 4.124 shows the sample image along with it's



(b) (c)

(d) (e)

Figure 4.121: Results of using the proposed method for registering the images shown in Figure 4.115 on the image shown in Figure 4.115–a using the proposed Newton enhanced method.

Table 4.26: Total distortion results when registering the images shown in Figure 4.115 with each other and themselves using the proposed Newton enhanced method.

First Image / **Second Image**	(a)	(b)	(c)	(d)	(e)
(a)	–	0.017%	0.046%	0.045%	0.062%
(b)	0.017%	–	0.049%	0.046%	0.061%
(c)	0.046%	0.049%	–	0.056%	0.066%
(d)	0.054%	0.055%	0.066%	–	0.088%
(e)	0.061%	0.061%	0.069%	0.070%	–

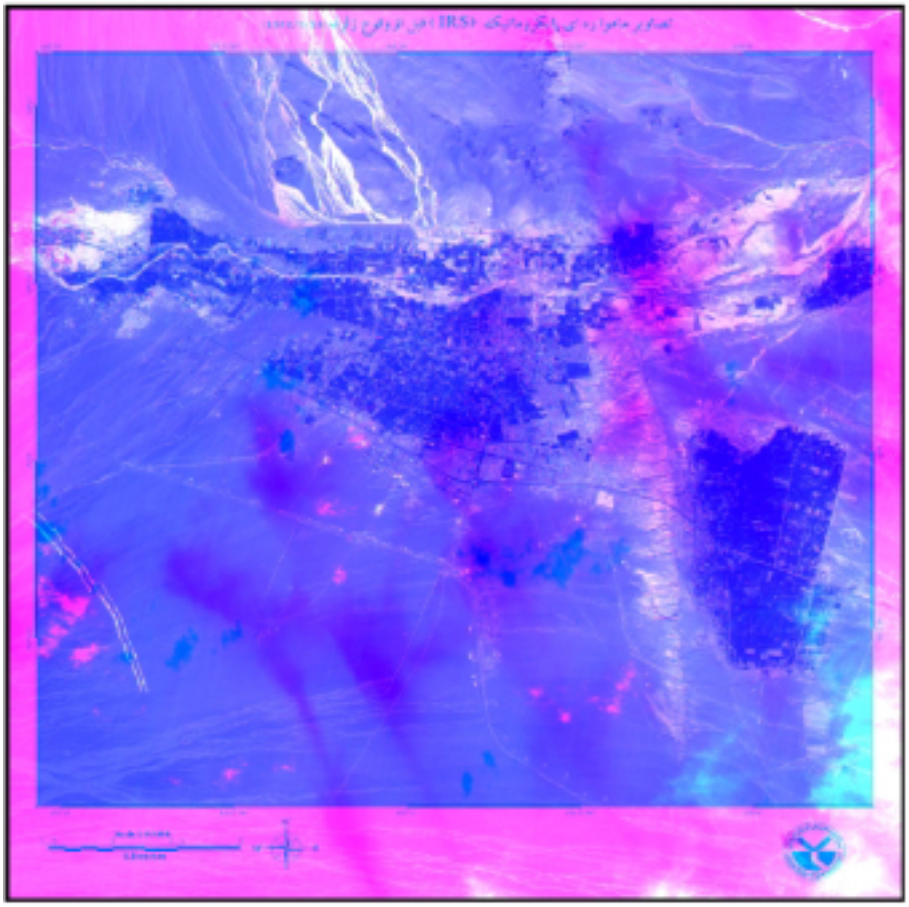


Figure 4.122: Result of applying the proposed Newton enhanced registration method on the images shown in figure 4.117.

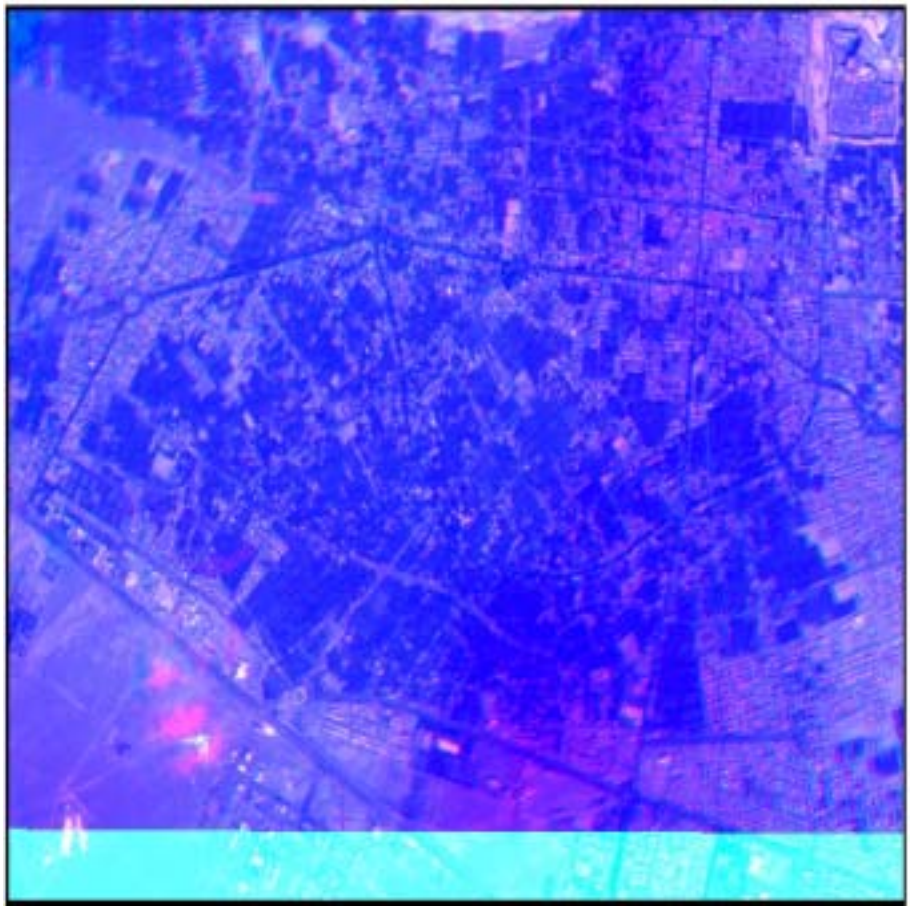


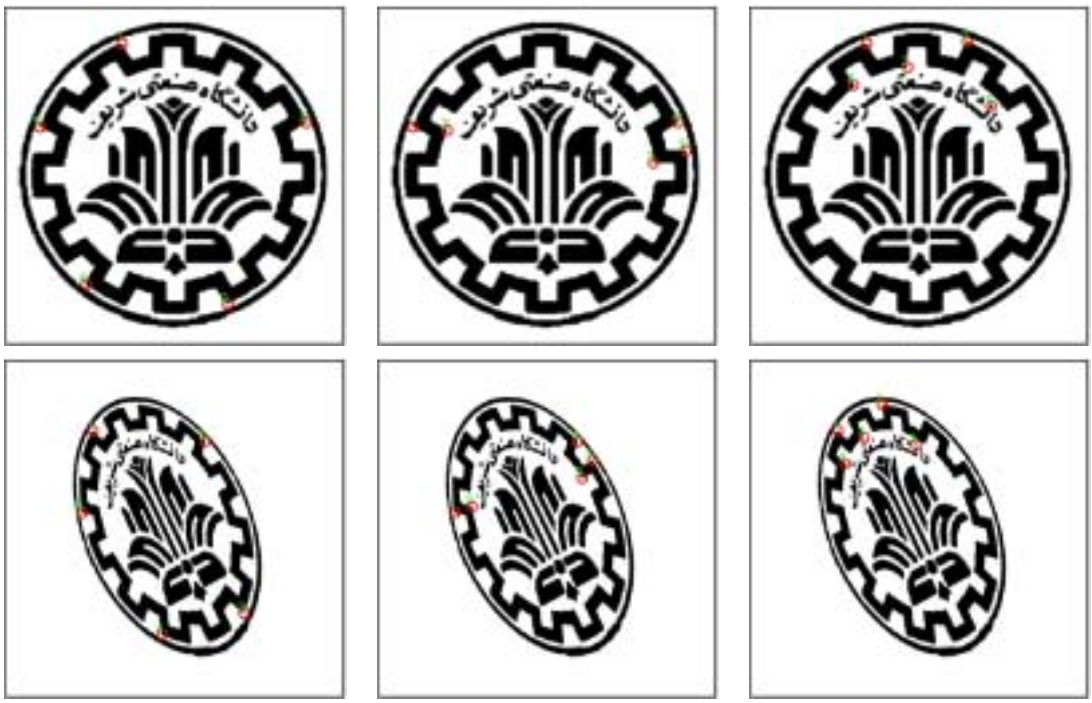
Figure 4.123: Result of applying the proposed Newton enhanced registration method on the images shown in figure 4.118.

(a) (b)

Figure 4.124: (a) Sample image. (b) Sample image undergoing scaling, rotation, translation, and a little skewness.

misaligned version. The image shown in Figure 4.124–b undergoes scaling, rotation, translation, and a little skewness. The aim of this experience is to evaluate three different patterns of control point distribution and their effects on compensating unwanted distortions. Figure 4.125 shows the three sets of control points. These patterns are called spread, linear, and zonal distributions.

Figure 4.126 shows the results of registering the two images of each set on each other. These results are acquired using the proposed Newton enhanced method. Table 4.27 lists the mismatching values. The mismatching value of two images is defined as the portion of pixels incorrectly registered. The actual function counts the number of black pixels registered into a white one and vice versa. These number is divided by the total number of pixels (512×512).

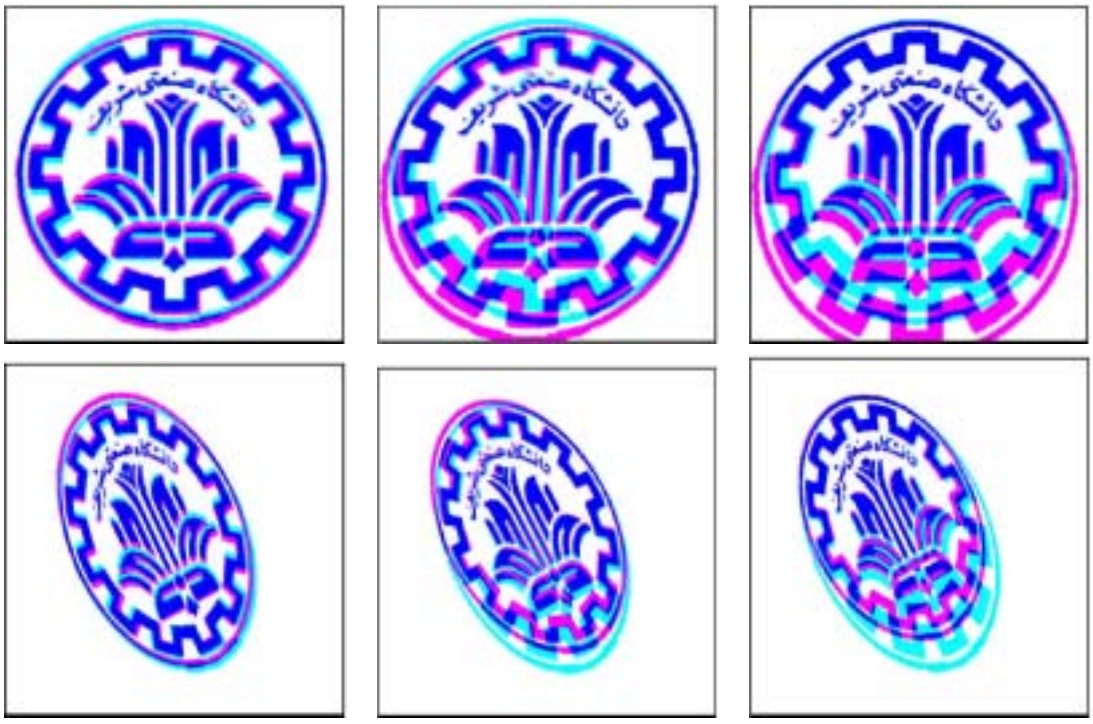


(a) (b) (c)

Figure 4.125: Three possible distribution patterns of control points. (a) Spread. (b) Linear. (c) Zonal.

Table 4.27: Mismatching value of the registering results shown in Figure 4.126.

Original Image	4.126–a	4.126–b	4.126–c	4.126–d	4.126–e	4.126–f
Second Image	4.126–b	4.126–a	4.126–d	4.126–c	4.126–f	4.126–e
Mismatching	27%	11%	36%	13%	41%	14%



(a) (b) (c)

Figure 4.126: Results of registering the images shown in Figure 4.125 on each other using the proposed Newton enhanced method. (a) Spread. (b) Linear. (c) Zonal.

Investigating Table 4.27 makes it clear that the best pattern of control points is to distribute them in the entire image area. This result is in complete agreement with common sense. Now, the doubtfulness of the robustness of registration approaches that depend on aligning small portions of the two images, like single curves (*e.g.*, see [383]) gets clear. We conclude that the feature detection method should be designed in the way that finds features all over the two images.

4.12 Content–based Pornography Filtering

4.12.1 Datasets

Using the *Google* advanced image search with enabled large size option and some erotic keywords, 3201 pornographic images were downloaded. Most of the images came from amateur weblogs devoted to pornography. All images are in color, represented in RGB color space and compressed using the standard *jpeg* compression with high visual quality. After the images are downloaded from the net, no preprocessing is performed on them. These images produce a dataset of erotic images. Both indoor and outdoor images of single and multiple people with both complex and simple backgrounds are collected in the dataset. It includes photographs of female Hollywood stars, professional photographs of erotic models, amateur pornographic images, and pictures taken in the beach.

When considering the web–pages containing pornographic content, two major categories are recognized. The first one is the group of professional agencies producing

and distributing erotic content. These homepages are mostly restricted to members and comply with regarding law. As described in [71] the main challenge of content–based pornographic image blocking is to put constraints on accidental access to adult content in the Internet. As such, no one may restrict adult motivated users from viewing pornographic content (that comply with the specialized law) [66]. Hence, the scope of this research does not include these professional sources of pornography. The second place to find adult images in the web, and the dominant one, is the category of homepages produced by amateurs, even teenagers. As the main challenge of the current research is to restrain these sources, all of the images in the erotic dataset are selected from free resources in the Internet. Note that while some of the images accompany copyright hints of professional agencies, they have been pirated and incorporated into free resources, and thus are added to the dataset.

the pornographic images in free homepages are separated into three major categories. Firstly, there are images that are transferred to the web directly from a digital camera. These images contain no further manipulations. Then, there come images that only contain a tag, indicating the source of the image. From the nature of the tags, it seems that they are added using one of the free batch image editing softwares available in the Internet. The last, and the least frequent, category is the few images containing professional manipulations. These images are those pirated from professional agencies. Figure 4.127–a shows the share of each category in the erotic dataset.

As seen here, almost all images in the erotic dataset are non–edited.

Figure 4.127–b shows the histogram of the width and the height of the images in the erotic dataset. As seen here, each histogram contains a single peak. As such, the width of the images is averagely 605 with the standard deviation of 196. Also, the height of the images is averagely 676 with the standard deviation of 235. The peaks of the width and height histograms occur in 520 and 685, respectively.

When it comes to the aspect ratio of the images, interesting features are found. For a $H \times W$ images, its corresponding aspect ratio is defined as $\min\{\frac{H}{W}, \frac{W}{H}\}$. In this way, we compensate for images captured horizontally, or vertically. Figure 4.127–c shows the histogram of the aspect ratios of the images in the erotic dataset. Hence, the aspect ratio of the images of the erotic dataset are mostly gathered about $\frac{2}{3}$ and $\frac{3}{4}$. As an example, in a digital Canon $A60$ camera, the available resolutions are 1600×1200, 1024×768, 640×480, corresponding. These figures all lead to the aspect ratio of $\frac{3}{4}$. This fact, empowers the assumption that the free pornographic images in the Internet are just copied from the camera and distributed without any further manipulations.

Another interesting feature of the collected erotic dataset is the compression ratio of the images. Firstly, it should be emphasized that except for the static and animated banners present in the erotic resources, the dominant file format in this area is the standard JPEG. As Figure 4.127–d shows, the images in the erotic dataset are

represented averagely by 1.45 bits per pixel with the standard deviation of 0.8 bits per pixel. Also, the peak of the histogram happens at 1.2 bits per pixel.

Figure 4.127–e illustrates the number of people present in the images of the erotic dataset. Hence, most of the images in the dataset include a single (female) model. Figure 4.127–f classifies the images into two groups in terms of the existence of at least one face in the images. This categorization is performed to see if face detection may help localizing naked people [194, 122]. Less than one fourth of the erotic images in the dataset contain no faces. This fact implies that using a face detection method may be beneficial.

Figure 4.127–g shows the separation of the images in the erotic dataset according to the sexual organs present in them. This figure shows that about half of the erotic images do not contain any such sexual organs. These images include all those with occluded or covered parts.

Another categorization of the erotic image dataset comes in Figure 4.127–h. This chart shows that about ninety percent of the images in the erotic dataset are indoor images compared to the minority of 11% outdoor ones.

The skin area of some of the images of the erotic dataset are cropped manually. The cropped images are then resized to 64×64 pixels. As such, a new training dataset of 314 samples is generated, which is called the skin dataset. Another dataset of 106 patches, each containing pixels corresponding to a single material is also produced.

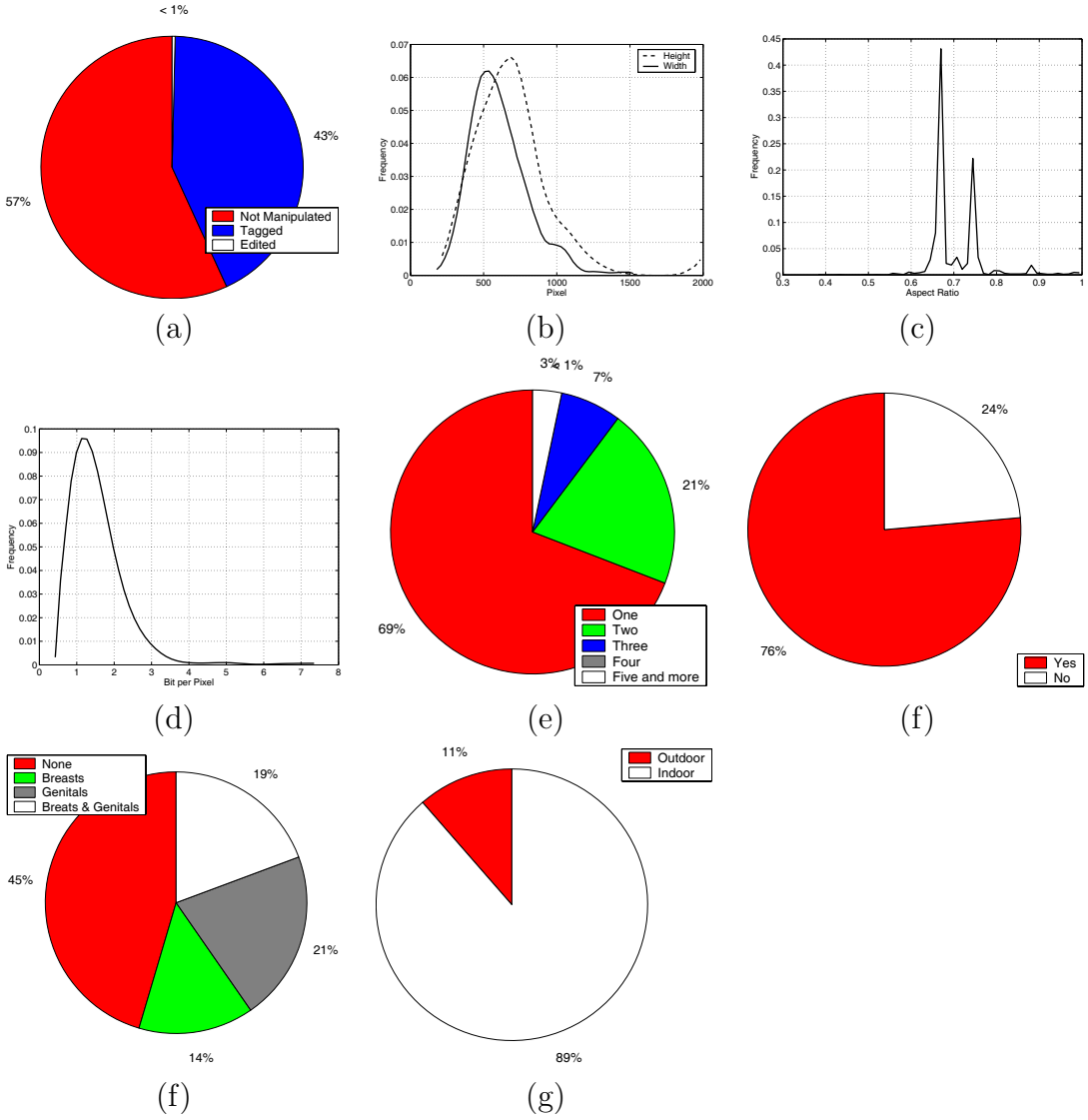


Figure 4.127: Statistics of the images in the erotic dataset. a) Classification of the images into three categories of not manipulated, tagged, and edited. b) Histogram of the width and the height of the images. c) Histogram of the aspect ratios. d) Histogram of bpp of the images. e) Number of people present in the images. f) Is there a face in the image? g) Separation of the images according to the visible sexual organs. h) Categorization in terms of being outdoor or indoor.

The patches are selected manually from images taken by a *Canon A60* digital camera at daylight with flash. This dataset is used as the control data. Figures 4.128 and 4.129 show the skin and control dataset, respectively. We emphasize the vast variety of colors related to skin swatches. Some colleagues of us were hardly believing some of the patches shown in Figure 4.128 to represent real skin samples. We believe that the main challenge for a proper skin detector is to adopt to all these skin samples, and yet reject non–skin tones. This is mainly caused by the practical situation in which a pornography–related skin detector should work; unconditional images just found somewhere in the Internet.

4.12.2 Look–up Table–Based Approach

Figures 4.130 and 4.131 show the ROC curves of the OLUT and BLUT–based skin detectors in the color spaces under examination, respectively.

The ROC curves of the OLUT–based skin detector for the 21 color spaces under examination (see Appendix 4.130) reveal several interesting features about the color spaces and their performance in OLUT–based skin detection. Firstly, selecting those color spaces resulting in points in the ROC, satisfying $TP > 90\%$ and $FP < 10\%$, results only in RGB, $HMMD$, $I_1I_2I_3$, I_2I_3, HSI, HSV, $HSV.HS$, VH, C_bC_r, YC_bC_r, IQ, $YUV.UV$, $CIE-Luv$, $CIE-uv$, Nbr, UV°, IQ°, $C_bC_r^\circ$, HSV^c, HS^c, TSL, TS. Thus, out of $21 * 7 = 147$ investigated color spaces, only 22 has shown preliminary acceptable results. In this way, if a color space is selected by chance, there is only

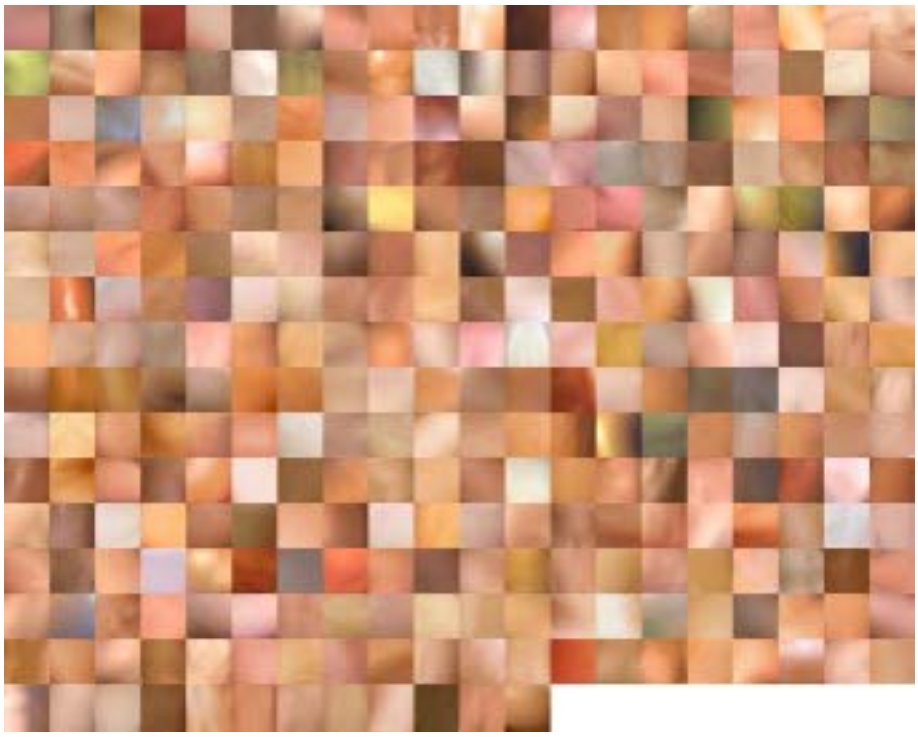
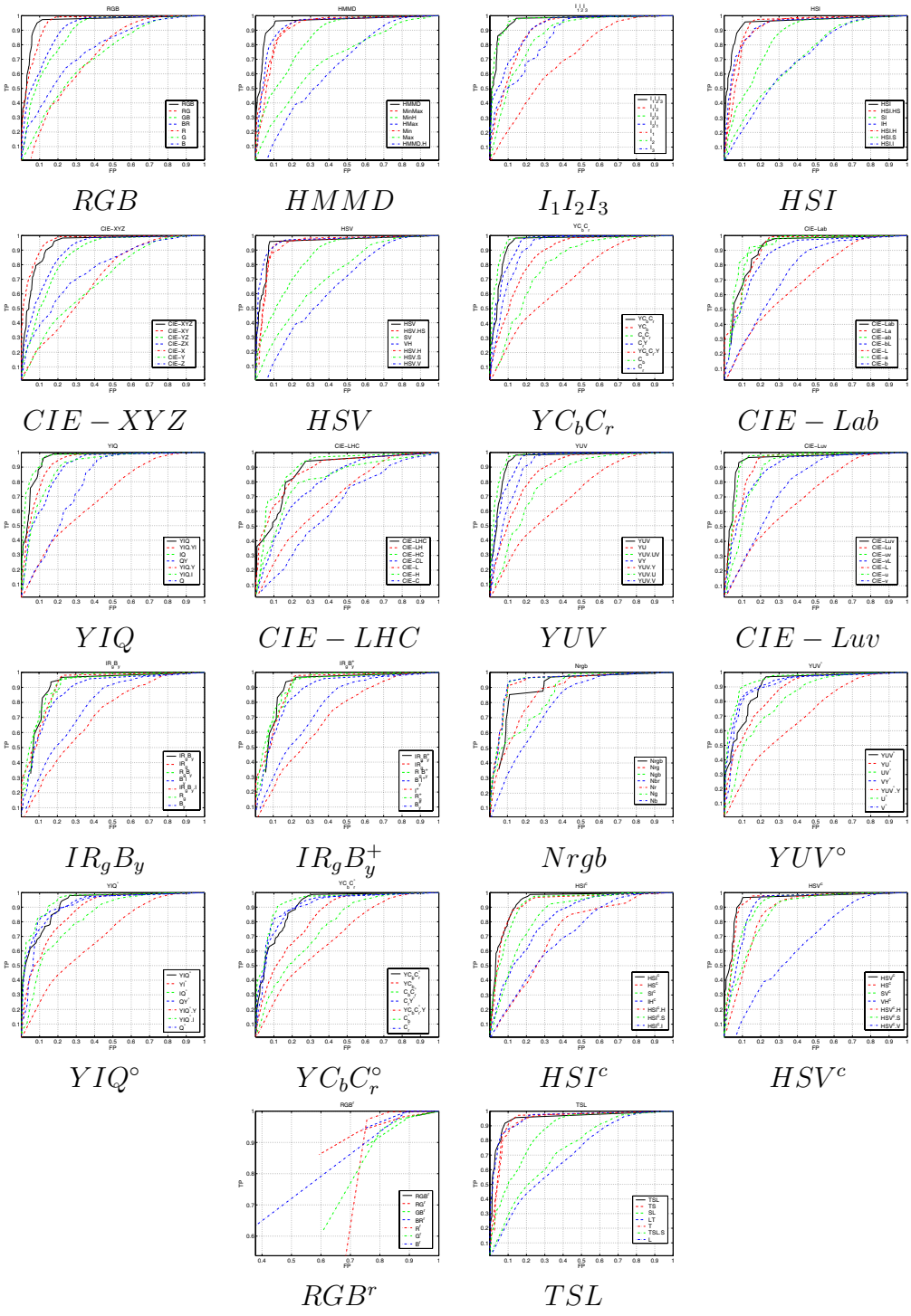


Figure 4.128: Skin Dataset.

Figure 4.129: Control Dataset.



RGB $HMMD$ $I_1I_2I_3$ HSI

$CIE-XYZ$ HSV YC_bC_r $CIE-Lab$

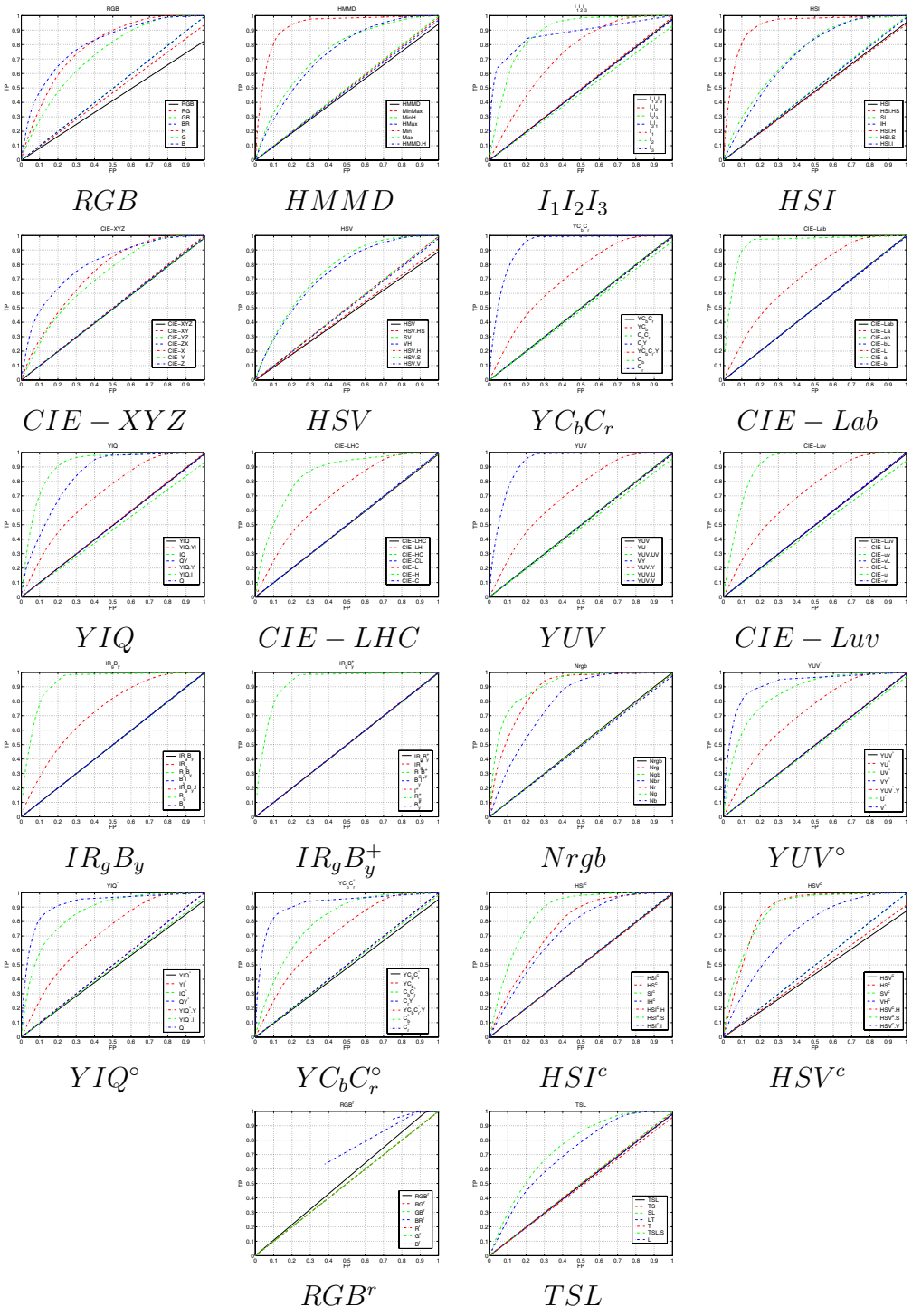
YIQ $CIE-LHC$ YUV $CIE-Luv$

IR_gB_y $IR_gB_y^+$ $Nrgb$ YUV°

YIQ° $YC_bC_r^\circ$ HSI^c HSV^c

RGB^r TSL

Figure 4.130: ROC curves of the OLUT–based skin detector.



RGB $HMMD$ $I_1I_2I_3$ HSI

$CIE-XYZ$ HSV YC_bC_r $CIE-Lab$

YIQ $CIE-LHC$ YUV $CIE-Luv$

IR_gB_y $IR_gB_y^+$ $Nrgb$ YUV°

YIQ° $YC_bC_r^\circ$ HSI^c HSV^c

RGB^r TSL

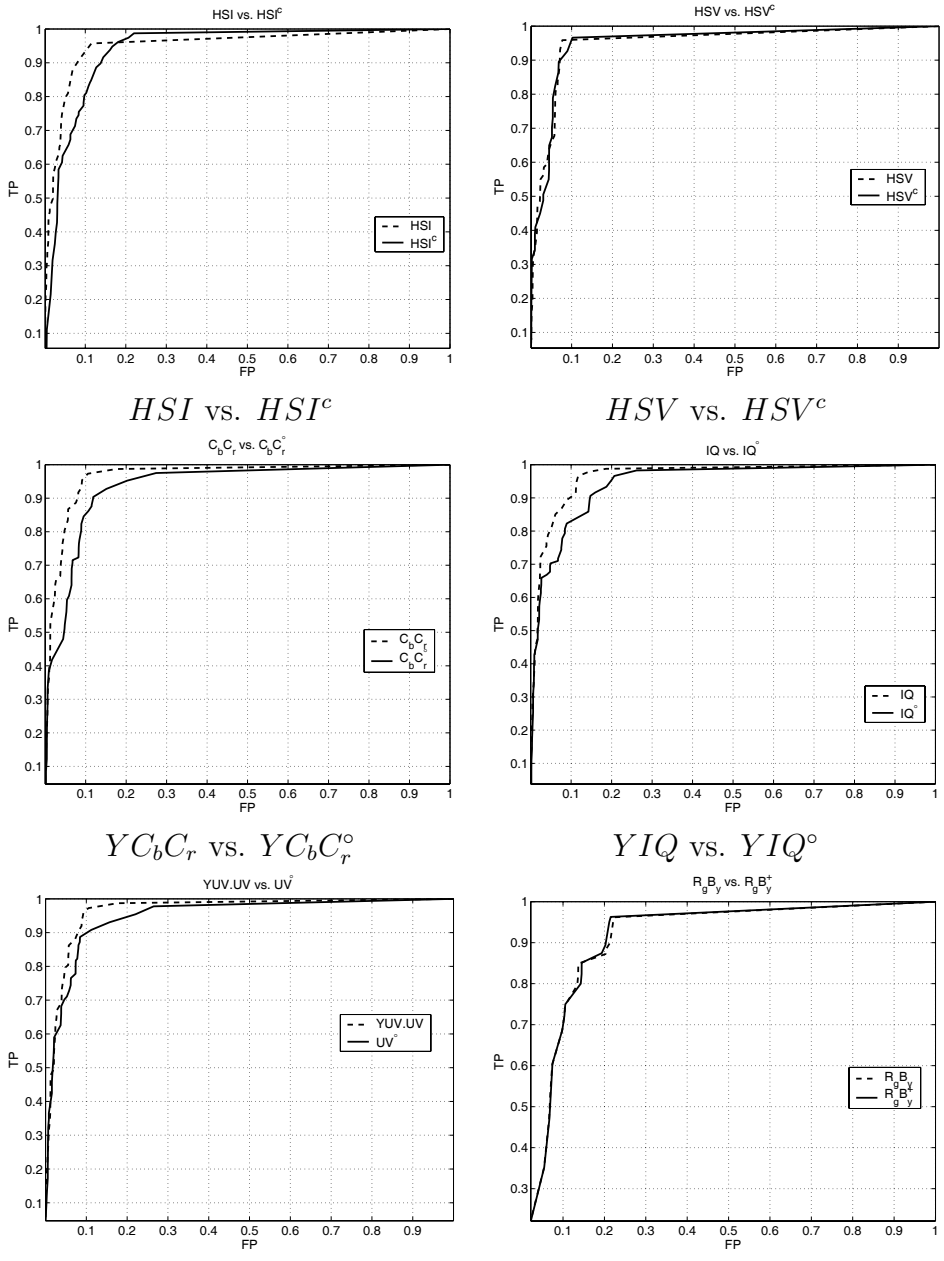
Figure 4.131: ROC curves of the BLUT–based skin detection.

15% hope that it will be helpful for OLUT–based skin detection. This fact proves that the ongoing investigation is worthy.

Another interesting outcome of the described experiment is the evaluation of the color spaces produced by altering elderly ones. There are six pairs of corresponding color spaces in this experiment. Namely, (HSI, HSI^c), (HSV, HSV^c), $(YC_bC_r, YC_bC_r^\circ)$, (YIQ, YIQ°), (YUV, YUV°), and $(IR_gB_y, IR_gB_y^+)$. Figure 4.132 compares the best achievements of the color spaces in each pair. These are (HSI, HSI^c), (HSV, HSV^c), $(C_bC_r, C_bC_r^\circ)$, (IQ, IQ°), $(YUV.UV, UV^\circ)$, and $(R_gB_y, R_gB_y^+)$. None of the manipulated color spaces outperform their regarding ancestors. In fact, the alterations in the structure of more classic color spaces have declined their performance.

An important result of this experiment is that in YUV, YIQ, and YC_bC_r, removing the illumination–related component (Y) increases the performance of skin detection. Thus, the definition of the illumination component in these three color spaces is a good step toward getting invariant to illumination.

Figure 4.133 illustrates the best ROC curves obtained from the color spaces in this experiment. We believe that the difference between the performance of these color spaces is not meaningful. Figure 4.134 shows the corresponding LUTs obtained in this experiment. Table 4.28 lists the resulting true and false positives. We emphasize that further comparison of these LUTs should be based on a larger dataset. Note that



HSI vs. HSI^c HSV vs. HSV^c

YC_bC_r vs. $YC_bC_r^\circ$ YIQ vs. YIQ°

YUV vs. YUV° IR_gB_y vs. $IR_gB_y^+$

Figure 4.132: Comparison of the original color spaces with their manipulated versions in terms the results of the best combination for LUT–based Skin Detection.

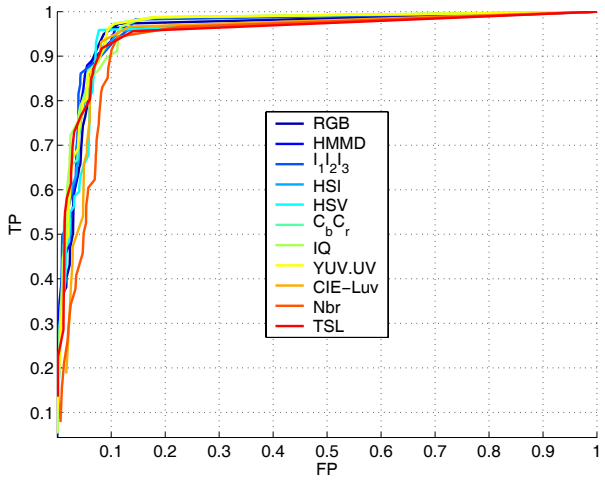


Figure 4.133: Best ROC curves for OLUT–based skin detection obtained from the investigated color spaces.

as the computation of the LUTs is performed using higher number of bins, there are some TP values less than the preselected margin of 90%. Here, we have used 32 and 256 bins for 3– and 2–D histograms, respectively, to acquire a more precise OLUT.

The skin map detection process is performed on the images in the erotic dataset, using each one of the 11 computed LUTs. Then, the results are separated into four categories of perfect, partial, excessive, and irrelevant. The perfect results are those, in which the skin detection process has carefully detected all skin pixels, with negligible false positives. The partial, and the excessive results are those with many false negative or many false positive pixels, respectively. Finally, irrelevant results are those in which the skin detection process has resulted completely unacceptably. Note that all these term are subjective definitions, and should be regarded carefully. Thus,

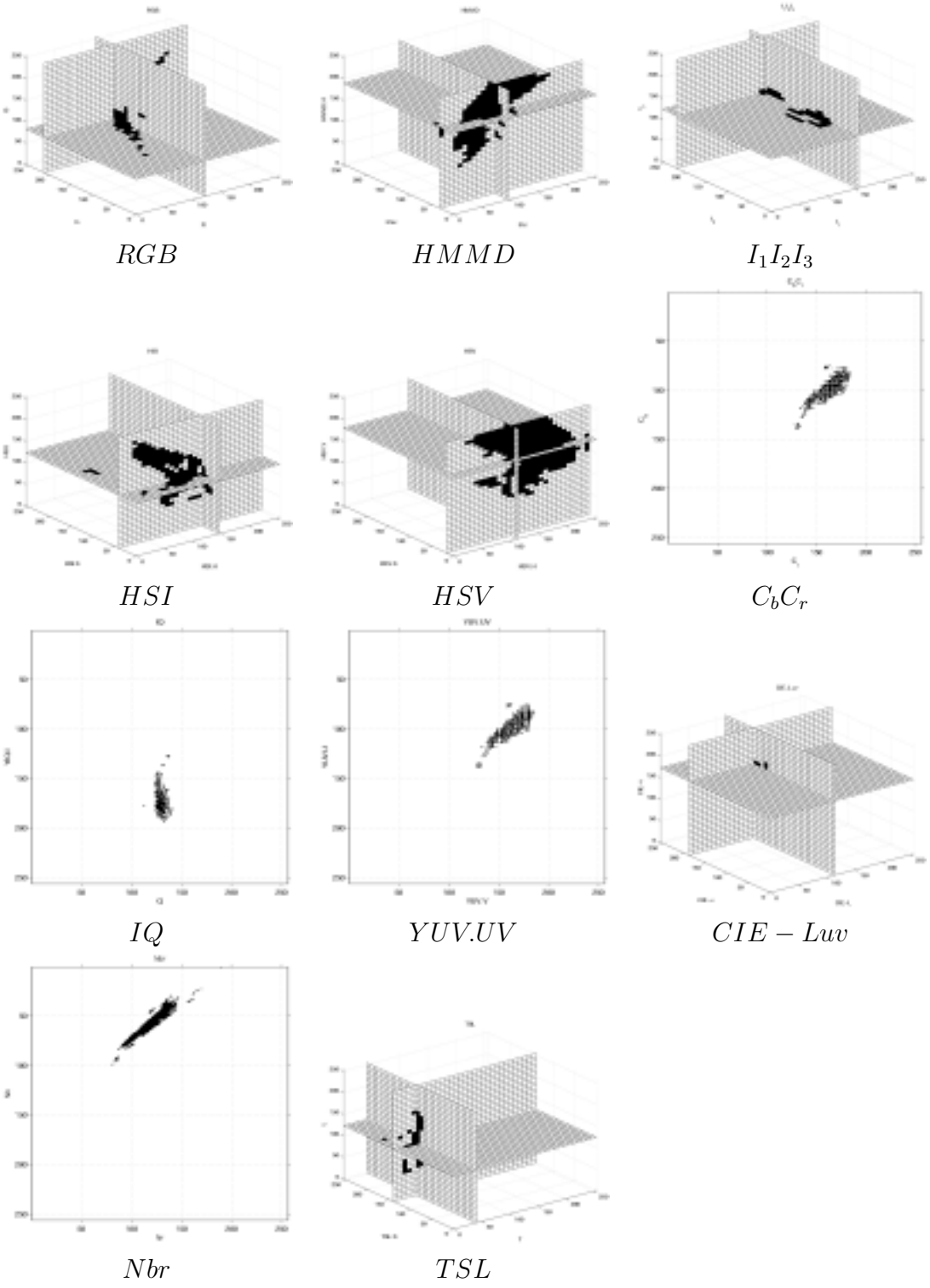


Figure 4.134: Best LUTs produced by the color spaces under investigation.

Table 4.28: False and true positive results of the best LUTs produced by the color spaces under investigation.

Color Space	True Positive	False Positive
RGB	91%	5.9%
$HMMD$	90%	6%
$I_1I_2I_3$	91%	6.3%
HSI	85%	5%
HSV	93%	5.8%
C_bC_r	89%	7.9%
IQ	76%	3.1%
$YUV.UV$	90%	8.5%
$CIE-Luv$	91%	7.2%
Nbr	86%	8.7%
TSL	91%	6.4%

the results of this test are only referable when there is a large difference between the portions of each category for two distinct LUTs. Figure 4.135 shows the results. This tests reveals that the performance of Nbr is absolutely better than others. The reader should be aware that while Nbr gives the best possible OLUTs in this experiment, it has only performed perfectly in 49% of the erotic images. We argue that the weakness of the OLUT-based approaches comes from its essential definition that neglects the histogram of non–skin samples. This inefficiency is carefully considered in Baysian approach, discussed in Section 3.11.1.

Selecting those color spaces capable of giving a BLUT satisfying $TP > 90\%$ and $FP < 20\%$, simultaneously, results in $CIE-a$, $CIE-u$, Min, $HSI.H$, R_g, R_g^+, C_r, C_r°, $YIQ.I$, Q°, $YUV.V$, and V° (see Appendix 4.130). Note that here we have doubled the margin for FP, as the ROC curves tend to fall in Baysian framework,

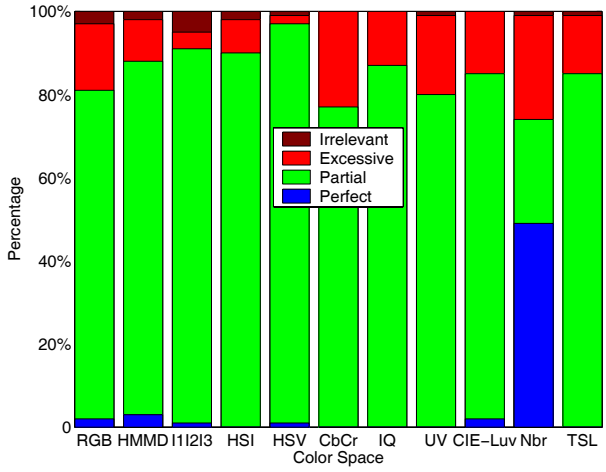


Figure 4.135: Results of computing the skin map using the best available LUTs.

compared to the OLUT–based one. Also, note that the ROC in $CIE - a$ is visibly outperforming others, even touching the $FP < 10\%$ margin. Note that selecting a color space by chance, there is only a 0.6% chance that it will yield proper BLUT–based skin detection results. Another interesting result of this experiment is that the best color spaces in Baysian framework are 1–D ones, compared to the case of OLUT–based skin detection in which the best eleven color spaces constitute eight 3–D, and three 2–D spaces and no 1–D color space.

Figure 4.136 compares the best achievements of the color spaces for BLUT–based skin detection in each pair of original and manipulated color spaces. There are six pairs of corresponding color spaces in this experiment. Namely, (HSI, HSI^c), (HSV, HSV^c), $(YC_bC_r, YC_bC_r^\circ)$, (YIQ, YIQ°), (YUV, YUV°), and $(IR_gB_y, IR_gB_y^+)$. The corresponding pairs of best combination of are $(HSI.H, H^c)$, $(HSV.S, H^c)$, (C_r, C_r°),

$(YIQ.I, Q^{\circ})$, $(YUV.V, UV^{\circ})$, and (R_g, R_g^+). As visible in Figure 4.136, the HSI^c performs weaker compared to the HSI. In contrast, the HSV^c is far more better than the HSV. The $YC_bC_r^{\circ}$, YIQ°, and YUV° does not have anything to offer compared to their respective ancestors. IR_gR_y and $IR_gR_y^+$ are giving completely the same ROC curves.

Figure 4.137 compares the ROC curves of the best color spaces for BLUT–based skin detection.

Figure 4.138 shows the best BLUTs produced by the color spaces under investigation. Table 4.29 lists the resulting true and false positives. We again emphasize that further comparison of these BLUTs should be based on a larger dataset. Note that as the computation of the BLUTs is performed using higher number of bins, there are some TP values less than the preselected margin of 90%. Here, we have used 32 and 256 bins for 3– and 2–D histograms, respectively, to acquire a more precise BLUT. Note that the general results of BLUT–based skin detection are worse than those of OLUT–based skin detection in the training set.

The images in the erotic dataset are processed using the computed BLUTs. Then the results are separated into the four groups of perfect, partial, excessive, and irrelevant. Figure 4.139 shows the results. It is visible that V° and C_r° are dominantly outperforming the others. Comparing Figure 4.139 with Figure 4.135 reveals that utilizing the Baysian approach results in a higher rate of perfect classification, while

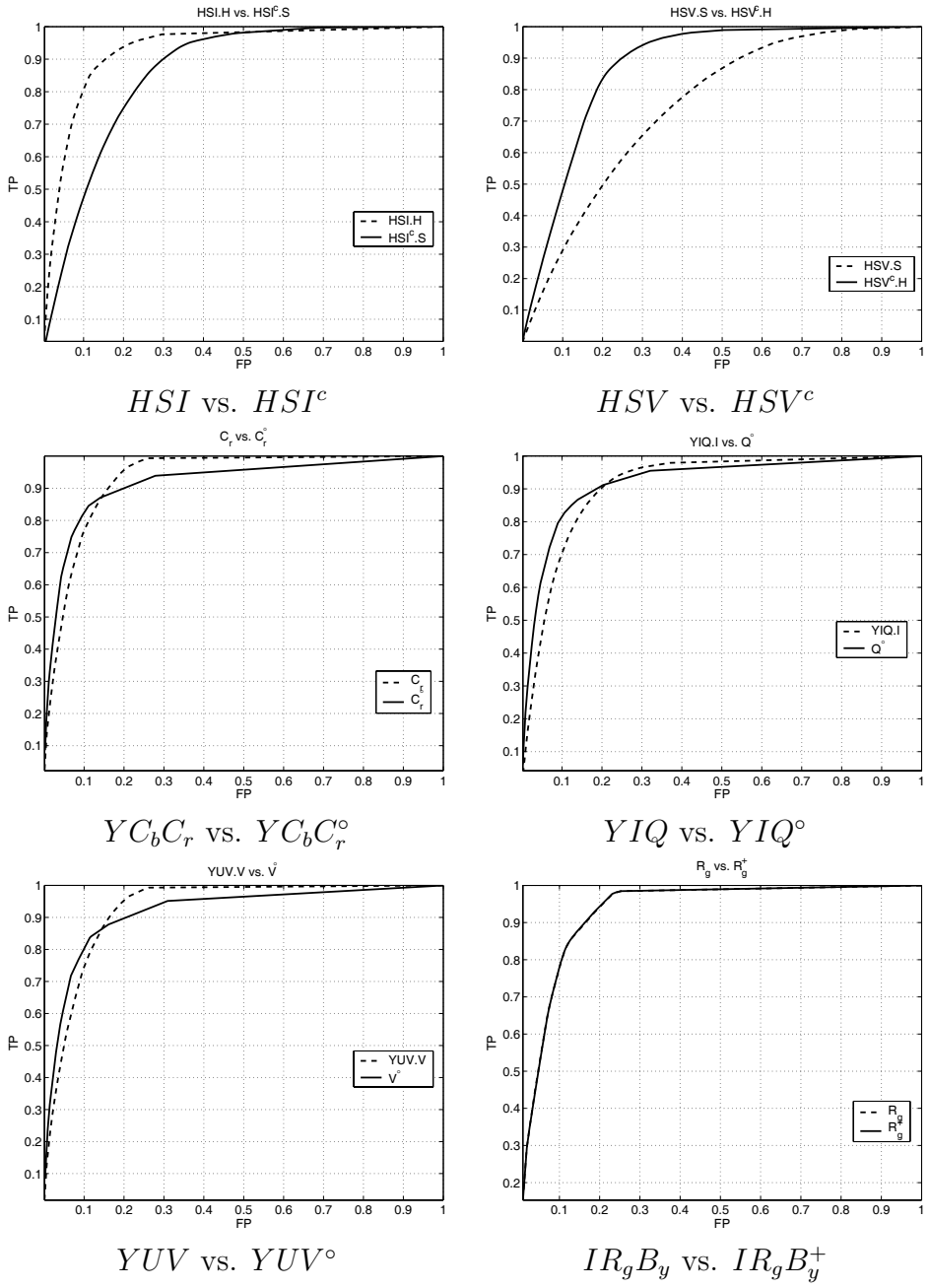


Figure 4.136: Comparison of the original color spaces with their manipulated versions in terms the results of the best combination for BLUT–based Skin Detection.

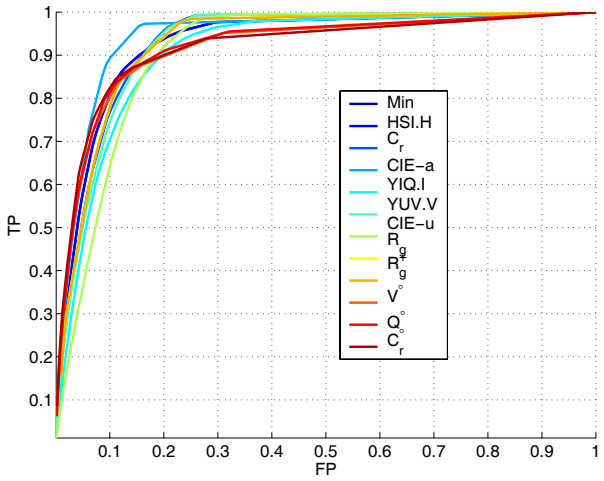


Figure 4.137: Best ROC curves for BLUT skin detector obtained from the investigated color spaces.

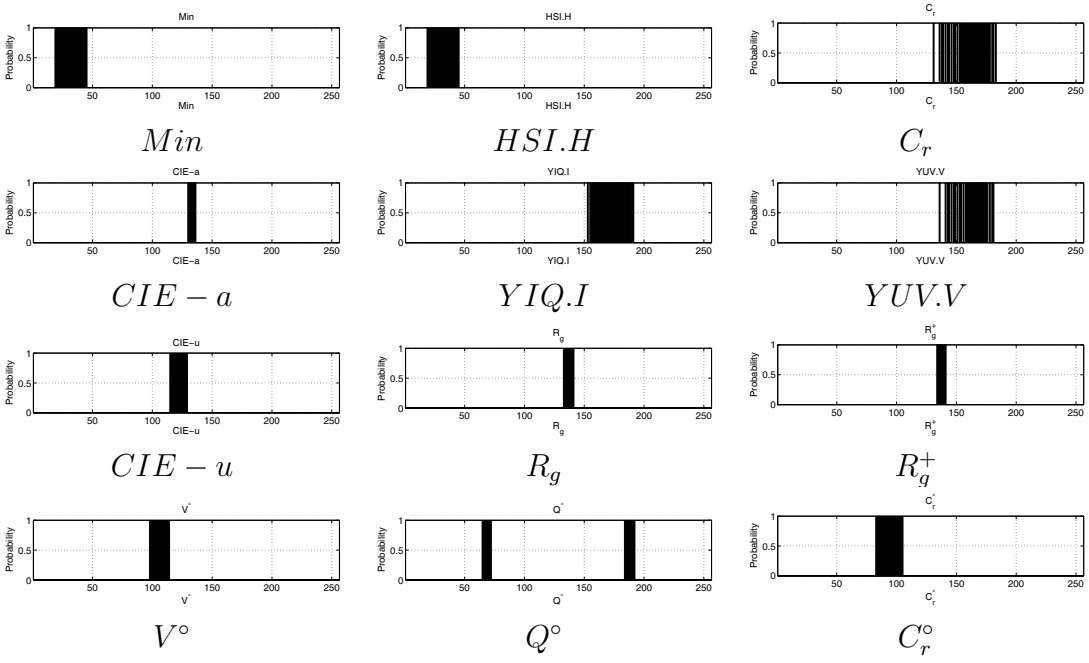


Figure 4.138: Best BLUTs produced by the color spaces under investigation.

Table 4.29: False and true positive results of the best BLUTs produced by the color spaces under investigation.

Color Space	True Positive	False Positive
Min	85%	13%
$HSI.H$	85%	13%
C_r	22%	93%
$CIE-a$	92%	14%
$YIQ.I$	83%	18%
$YUV.V$	89%	18%
$CIE-u$	57%	11%
R_g	78%	16%
R_g^+	74%	13%
V°	92%	28%
Q°	92%	28%
C_r°	95%	32%

it also increases the possibility of irrelevant vectors to be classified as skin. We argue that this occurs when both the nominator ($P(skin|\vec{c})$) and denominator ($P(\sim skin|\vec{c})$) of the fraction $p^+(\vec{c})$ get zero, resulting in an non–meaningful value. This fault may be resolved using a better control dataset (which includes more non–skin samples).

4.12.3 Histogram–based Approach

The χ^2 distance is computed between each two sample in each dataset, in twelve 3–D spaces corresponding to the 12 color spaces under investigation, in 36 two dimensional spaces corresponding to the spaces resulting from rejecting each one of the components in each one of the color spaces, and in 36 one dimensional color spaces, corresponding to single components of the color spaces. All histograms are computed in 16 bins. Tables 4.30, 4.31, 4.32, and 4.33 list the regarding values of expectation and standard

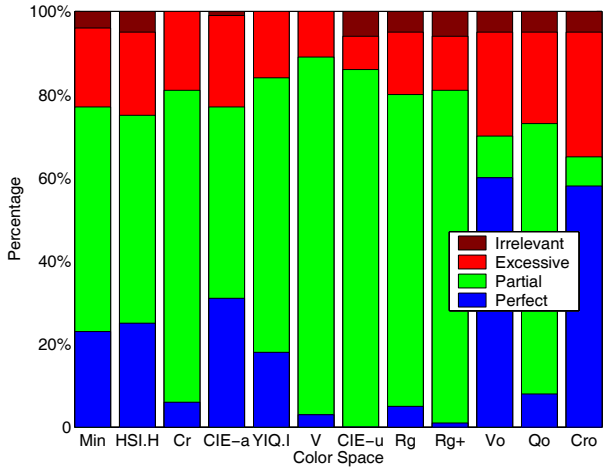


Figure 4.139: Results of computing the skin map using the best available BLUTs.

deviation of χ^2 in the skin dataset and the control dataset, respectively.

Note that, the mathematical expectation of χ^2 distance in all 3–D histograms listed in Tables 4.30 and 4.31 are more than 0.65. Thus, none of the investigated color spaces are proper choices for filtering skin tone in their 3–D representation. Similarly, rejecting one component does not help much. As shown in Tables 4.30 and 4.31, all χ^2 distances for 2–D histograms, except for the case of rejecting L in $CIE - Lab$, are more than 50. While, the later case results in average distance of about 0.33 with standard deviation of about 0.33, meaning a range of distances between zero and 0.66, which is not practical. The results on the 2–D histograms in the skin dataset highlight an interesting feature, when searching for the best case for rejecting one component of each color spaces (see Table 4.34). Note that, except for R in RGB and Z in $CIE - XYZ$, all other components listed in Table 4.34,

Table 4.30: Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the skin dataset in different color spaces. Part I.

Color Space	1–D					
	c_1		c_2		c_3	
	η	σ	η	σ	η	σ
RGB	0.61	0.31	0.55	0.30	0.52	0.31
$HMMD$	0.34	0.32	0.52	0.31	0.61	0.31
$I_1I_2I_3$	0.55	0.30	0.55	0.34	0.15	0.23
HSI	0.34	0.32	0.56	0.32	0.55	0.30
$CIE-XYZ$	0.57	0.31	0.56	0.31	0.52	0.31
HSV	0.14	0.25	0.55	0.32	0.61	0.31
YC_bC_r	0.56	0.31	0.46	0.36	0.45	0.34
$CIE-Lab$	0.51	0.32	0.00	0.01	0.33	0.33
YIQ	0.56	0.31	0.51	0.35	0.27	0.27
$CIE-LHC$	0.51	0.32	0.29	0.32	0.37	0.33
YUV	0.56	0.31	0.46	0.36	0.45	0.34
$CIE-Luv$	0.51	0.32	0.41	0.33	0.44	0.36

Table 4.31: Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the skin dataset in different color spaces. Part II.

Color Space	2–D						3–D	
	c_1,c_2		c_2,c_3		c_3,c_1			
	η	σ	η	σ	η	σ	η	σ
RGB	0.80	0.23	0.75	0.25	0.84	0.21	0.89	0.17
$HMMD$	0.67	0.28	0.84	0.21	0.73	0.27	0.88	0.18
$I_1I_2I_3$	0.81	0.24	0.61	0.33	0.61	0.30	0.83	0.22
HSI	0.67	0.29	0.83	0.21	0.69	0.28	0.87	0.18
$CIE-XYZ$	0.64	0.27	0.77	0.24	0.80	0.23	0.82	0.21
HSV	0.59	0.32	0.86	0.19	0.65	0.30	0.87	0.19
YC_bC_r	0.76	0.26	0.62	0.32	0.75	0.26	0.84	0.21
$CIE-Lab$	0.51	0.32	0.33	0.33	0.68	0.29	0.68	0.28
YIQ	0.77	0.25	0.63	0.30	0.69	0.26	0.84	0.20
$CIE-LHC$	0.65	0.30	0.51	0.33	0.69	0.28	0.75	0.25
YUV	0.76	0.26	0.62	0.32	0.75	0.26	0.84	0.21
$CIE-Luv$	0.70	0.28	0.60	0.32	0.73	0.28	0.80	0.23

Table 4.32: Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the control dataset in different color spaces. Part I

Color Space	1–D					
	c_1		c_2		c_3	
	η	σ	η	σ	η	σ
RGB	0.76	0.30	0.70	0.30	0.72	0.31
$HMMD$	0.83	0.27	0.67	0.31	0.73	0.30
$I_1I_2I_3$	0.71	0.30	0.79	0.31	0.71	0.36
HSI	0.83	0.27	0.71	0.30	0.71	0.30
$CIE-XYZ$	0.72	0.30	0.70	0.30	0.72	0.31
HSV	0.84	0.28	0.72	0.30	0.73	0.30
YC_bC_r	0.70	0.30	0.73	0.34	0.78	0.35
$CIE-Lab$	0.67	0.31	0.53	0.43	0.62	0.36
YIQ	0.70	0.30	0.78	0.33	0.70	0.36
$CIE-LHC$	0.67	0.31	0.78	0.30	0.58	0.37
YUV	0.70	0.30	0.73	0.34	0.78	0.35
$CIE-Luv$	0.67	0.31	0.76	0.34	0.70	0.34

Table 4.33: Expectation, η, and standard deviation, σ, of χ^2 distance between samples in the control dataset in different color spaces. Part II

Color Space	2–D						3–D	
	c_1,c_2		c_2,c_3		c_3,c_1			
	η	σ	η	σ	η	σ	η	σ
RGB	0.94	0.14	0.91	0.17	0.93	0.15	0.98	0.09
$HMMD$	0.94	0.15	0.92	0.16	0.95	0.13	0.98	0.08
$I_1I_2I_3$	0.92	0.17	0.92	0.19	0.91	0.18	0.97	0.11
HSI	0.94	0.15	0.92	0.16	0.94	0.15	0.98	0.08
$CIE-XYZ$	0.90	0.19	0.90	0.18	0.92	0.17	0.96	0.12
HSV	0.94	0.16	0.93	0.15	0.94	0.15	0.98	0.08
YC_bC_r	0.89	0.20	0.92	0.21	0.93	0.18	0.96	0.12
$CIE-Lab$	0.84	0.25	0.82	0.28	0.86	0.21	0.93	0.16
YIQ	0.93	0.17	0.91	0.20	0.91	0.17	0.96	0.11
$CIE-LHC$	0.93	0.16	0.89	0.23	0.86	0.21	0.96	0.11
YUV	0.89	0.20	0.92	0.21	0.93	0.18	0.96	0.12
$CIE-Luv$	0.92	0.18	0.91	0.21	0.89	0.19	0.97	0.11

Table 4.34: Best selection for rejecting one component of the color spaces for skin tone filtering.

Color Space	**Component**
RGB	R
$HMMD$	Max
$I_1I_2I_3$	I_1
HSI	$HSI.I$
$CIE-XYZ$	$CIE-Z$
HSV	$HSV.V$
YC_bC_r	$YC_bC_r.Y$
YIQ	$YIQ.Y$
$CIE-LHC$	$CIE-L$
YUV	$YUV.Y$
$CIE-Luv$	$CIE-L$

are the illumination–related component. This is in accordance with the proposal of *Shin et. al.* that, rejecting the illumination component enhances the skin filtering results [323]. Note that, YC_bC_r is not showing any desirable results, in contrast with the general assumption and the *Shin et. al.*'s conclusion [323].

Tables 4.30 and 4.31 show that, single component histograms reveal the skin color information better than double and triple combinations. In fact, selecting the χ^2 expectation values below 0.20 in Tables4.30 and 4.31, $HSV.H$, $CIE-a$, and I_3 are the only choices for skin tone filtering, while, the high standard deviation of χ^2 distances in $HSV.H$, ($\sigma = 0.25$), and I_3, ($\sigma = 0.23$), makes their performance doubtful.

Tables 4.32 and 4.33 make the skin detection problem more clear. While the χ^2 distance between skin samples is giving it best results in I_3, $HSV.H$, and $CIE-a$,

Tables 4.32 and 4.33 show that the distance between irrelevant color tones in these spaces are adequately large. In fact, the χ^2 distance between different objects are 0.71 in I_3, 0.84 in $HSV.H$, and 0.53 in $CIE-a$. Hence, the best choices for skin tone detection are neither 3–D spaces, nor 2–D ones, but the 1–D spaces corresponding to $CIE-a$, $HSV.H$, and I_3, respectively.

To investigate the separability of skin tone in the above selected color components, ($CIE-a$, $HSV.H$, and I_3), the expectation value and the standard deviation of each sample in the skin dataset and the control dataset, in these three color components are computed and the resulting histograms are drawn (see Figure 4.140). In this figure, in each case, the left chart shows the histogram of average value of the color component, while the right chart shows the histogram of the standard deviation of the color component, both computed in a single dataset. The top row in Figure 4.140 corresponds to the control dataset, while the bottom row shows the results in the skin dataset. A color component will be a practical one for skin detection, if its values in the skin dataset varies completely with its values in the control dataset. Desperately, unless the good χ^2 results of $CIE-a$, the expectation histogram of $CIE-a$ in both datasets occupies the same range, $(125\cdots140)$. In the same way, I_3 and $HSV.H$ are also rejected. Thus, no combination of the components in the twelve color spaces under investigation in this thesis, are good tools for skin detection, when using thresholding, (as commonly used in the literature).

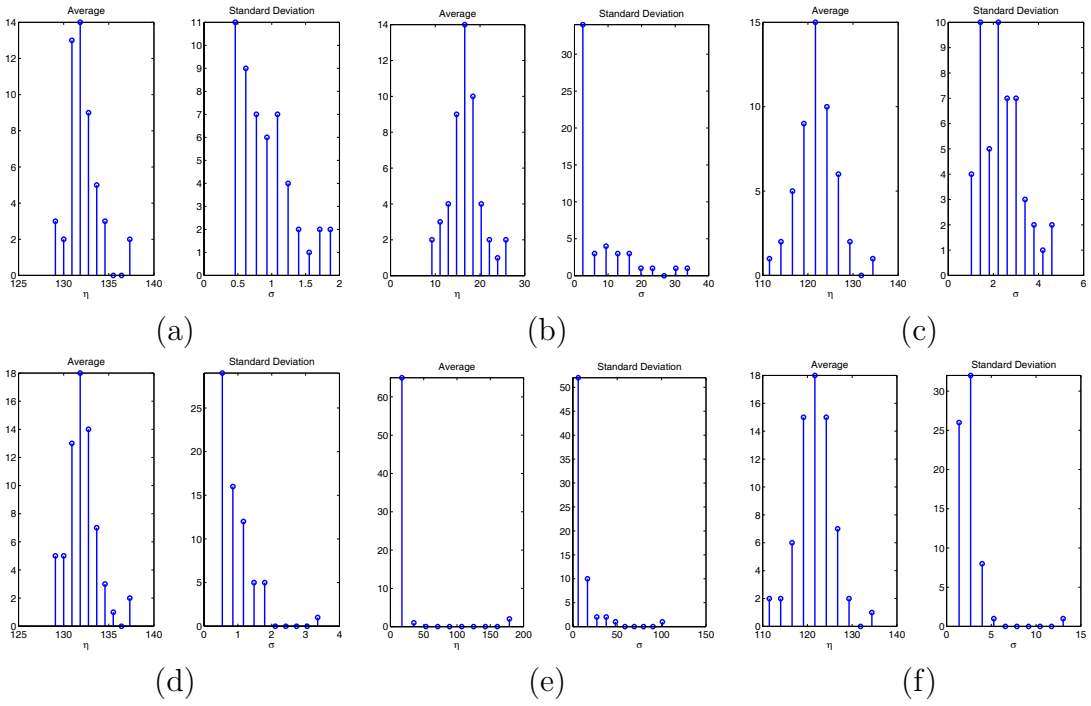


Figure 4.140: Histogram of the expectation and standard deviation of the sample images in the selected color components. (a),(d) $CIE - a$. (b),(e) $HSV.H$. (c),(f) I_3. (a),(b),(c) Control dataset. (d),(e),(f) Skin dataset.

Another test is performed to see whether the signature of the skin in the three color components of $CIE-a$, $HSV.H$, and I_3 are proper descriptors for skin tone filtering. The signatures are extracted from the skin dataset by averaging the 128-bin histograms of all samples in the respective color components. To rank a given patch in terms of complying with the signature, the χ^2 distance between its 128-bin histogram and the signature are computed. Figure 4.141 shows the results of classifying the samples of both datasets of skin, (top), and control, (bottom), into two classes of skin, (right), and non–skin, (left). Those samples are regarded as skin, which have χ^2 distance of less than 0.33 to the signature. The top row in Figure 4.141 shows the skin signature in the three color components of $CIE-a$, $HSV.H$, and I_3. Investigating Figure 4.141 shows that, while all three color components yield applicable results, $CIE-a$ is performing negligibly better. The performance of the signature method should be verified on a larger database.

4.12.4 PCA–based Approach

Figure 4.142 shows the ROC curve of the proposed skin detection method. Using this curve, the optimal threshold is selected. As such, using $\theta = 2042$ results in a true positive rate of 92% and a false positive rate of 22%. Figure 4.143 shows the principal skin color (the nearest point to all skin samples) computed in this work.

Figure 4.144 shows 24 sample images taken by an *A*60 *Canon* digital camera. All images are 512×512 and compressed with quality of 100 using the standard JPEG

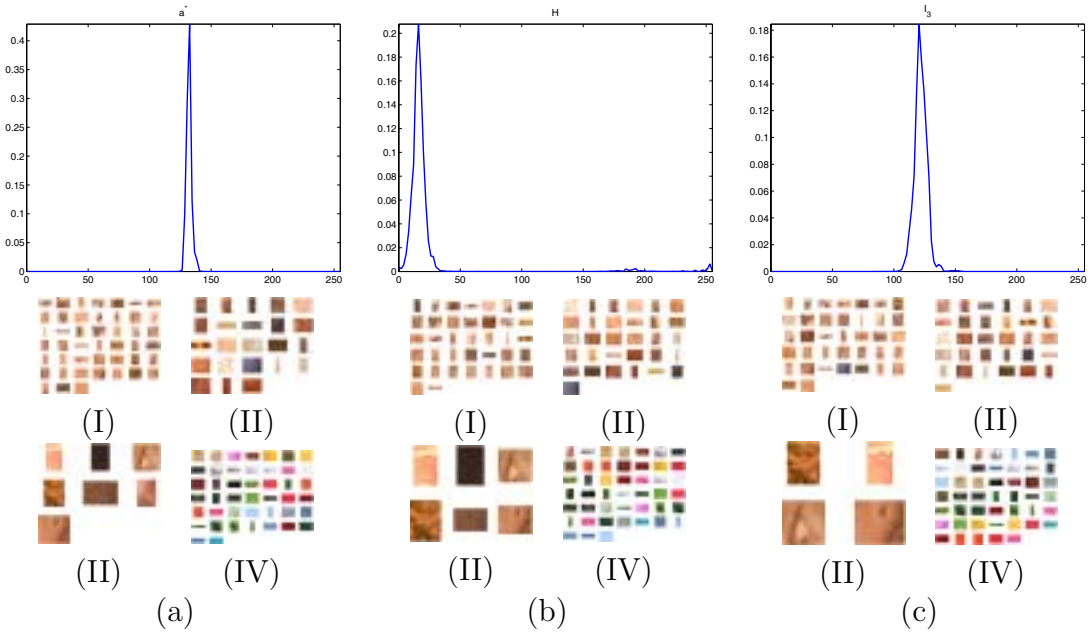


Figure 4.141: Results of classifying the samples in the two datasets of skin, ((I),(II)), and control, ((III),(IV)), into two classes of skin, ((I),(III)), and non–skin, ((II),(IV)). The top row shows the skin signature. The operation is performed in the three color components of (a) $CIE-a$, (b) $HSV.H$, and (c) I_3.

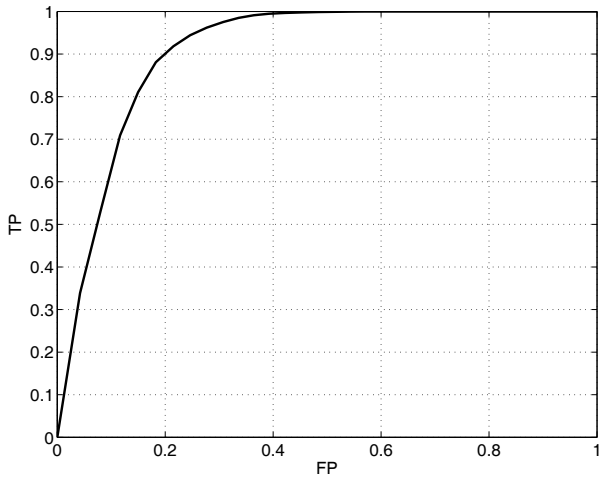


Figure 4.142: ROC curve of the proposed skin detection method.

Figure 4.143: Principal skin color computed by the proposed method.

format. The images are given to the skin detection procedure. Figure 4.145 shows the results. In each image, the non–skin portion of the image is shown in white, while for the skin pixels the original colors are saved.

While the sample images shown in Figure 4.144 are taken in a vast set of lighting conditions, the results of the proposed skin detection method are surprisingly promising. Although, in some samples there are non–skin pixels in the skin–map, most of them are reasonably skin–like. For example, the wall in Figures 4.145–b, c, i, and r and the soil in Figure 4.145–u are even misclassified by a human observer to be skin, when the images are magnified to only contain the respective portions.

An experiment is conducted to estimate the performance of the proposed method in finding the skin area in the erotic samples. The results show that in 81% of the images, the skin area is completely detected, with negligible irrelevant points assumed to be skin–related. While in 9% of the images the skin area is partially detected, there were about 10% of the images in which the detected area contained non–skin portions of the images. Another experiment carried out on the same dataset reveals that using

Figure 4.144: Sample images.

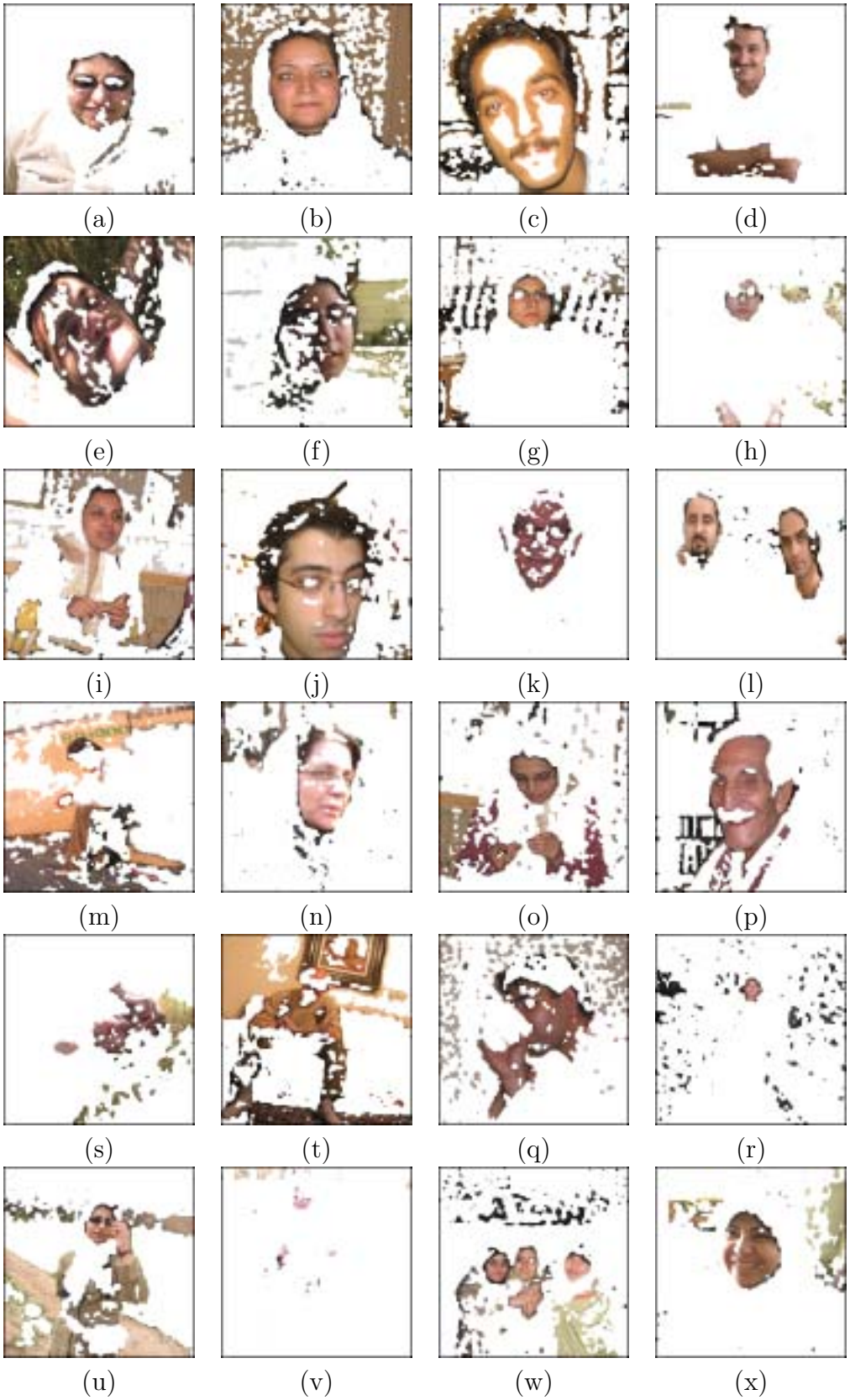


Figure 4.145: Results of applying the proposed skin detection method on the images shown in Figure 4.144.

Baysian inference, the highest rate for detecting the skin area in the erotic dataset is less than 60%. Using ordinary look–up table computation, this figure drops to 50% [38].

Here, we compare the proposed method with the conventional pixel–based method used in the pornography filtering literature. The Baysian and ordinary look–up table–based methods need the computation of the representation of the given image in a fixed color space. This averagely costs 8.7 ± 5.3 flops. Hence, using the look–up table based methods the process demands about $9WH$ flops to compute the skin map. Using the proposed method, the process should perform the PCA and the LPRE distance on a $\lambda \times \lambda$ patch. As such, $6\lambda^2 + 76$ flops are needed for each patch, resulting in $\frac{WH}{\lambda^2}(6\lambda^2 + 76)$ flops for the entire image. For the typical value of $\lambda = 8$, this figure equals $7WH$ flops. Hence, using nearest neighborhood interpolation the proposed method is about 30% faster than the pixel–based method. Using bilinear upsampling, about $\frac{8}{3}$ flops more are needed for each pixel, increasing the total figure to 9.7 flops. As such, using bilinear upsampling the proposed method is only 8% slower than the pixel–based method. We believe that the high performance of the proposed method compensate this lower speed.

4.13 Video Segmentation and Change Detection

Figure 4.146 shows the color image sequences used in this study. All images are non–compressed TIF files in the RGB color space. Table 4.35 lists the regarding image

Table 4.35: Image size, frame count, frame rejection basis (τ), cluster number (c), initialization time (t_0), and frame processing time (t_f) for the image sequences shown in Figure 4.146.

Name	**Size**	**Frames**	τ	c	t_0	t_f
Claire	208×240	10	1	3	6.5s	1.14s
Flower Garden	240×352	9	1	4	33.8s	2.56s
Foreman	288×352	250	5	4	64.5s	3.11s
Car Phone	144×176	201	5	3	35.9s	0.57s
Miss America	144×176	150	3	3	11.36s	0.63s
Table Tennis	240×352	21	1	3	51.0s	1.9s
Baseball	160×112	170	5	3	4.5s	0.28s

size and frame counts.

Figures 4.147, 4.148, 4.149, 4.150, 4.151, and 4.152 show the results of the proposed image segmentation and fuzzy change detection method performed on *Claire*, *Flower Garden*, *Foreman*, *Car Phone*, *Miss America*, and *Table Tennis* image sequences, respectively. Table 4.35 lists the respective values of τ, c, t_0, and t_f. Each figure contains a few sets of images separated by horizontal lines. In each set, the first row shows the original images, the second row shows the defuzzyfied segmentation (using maximum likelihood), the third row shows the fuzzy change detection masks and the fourth row indicates the frame indices. The performance of the proposed method is visible in these examples.

Figure 4.153 illustrates a typical respond of the proposed method to the scene cut. While in almost all of the available methods, when a scene cut happens, the algorithm has to initialize the whole process, the proposed method tolerates scene change. Figure 4.153 shows *Baseball* image sequence where a dissolve occurs in the

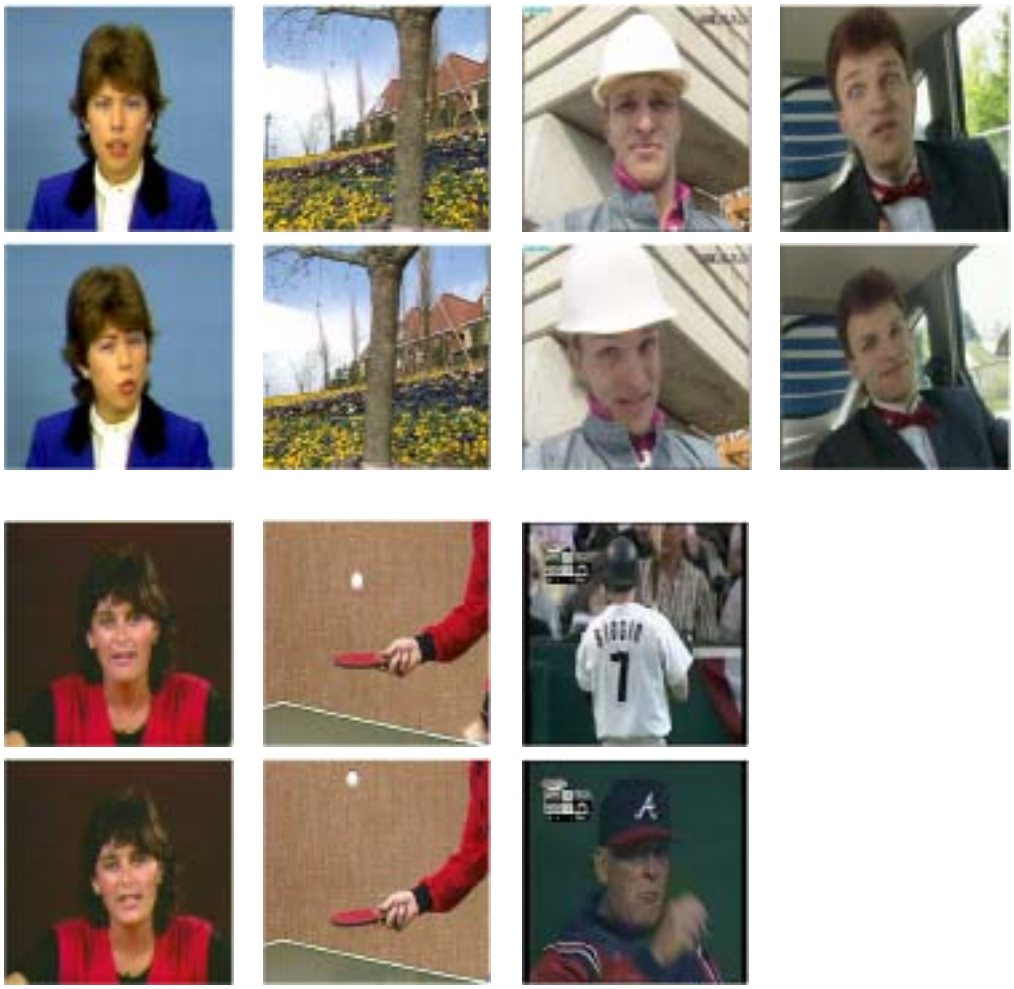


Figure 4.146: Some test image sequences (see Table 4.35 for details).

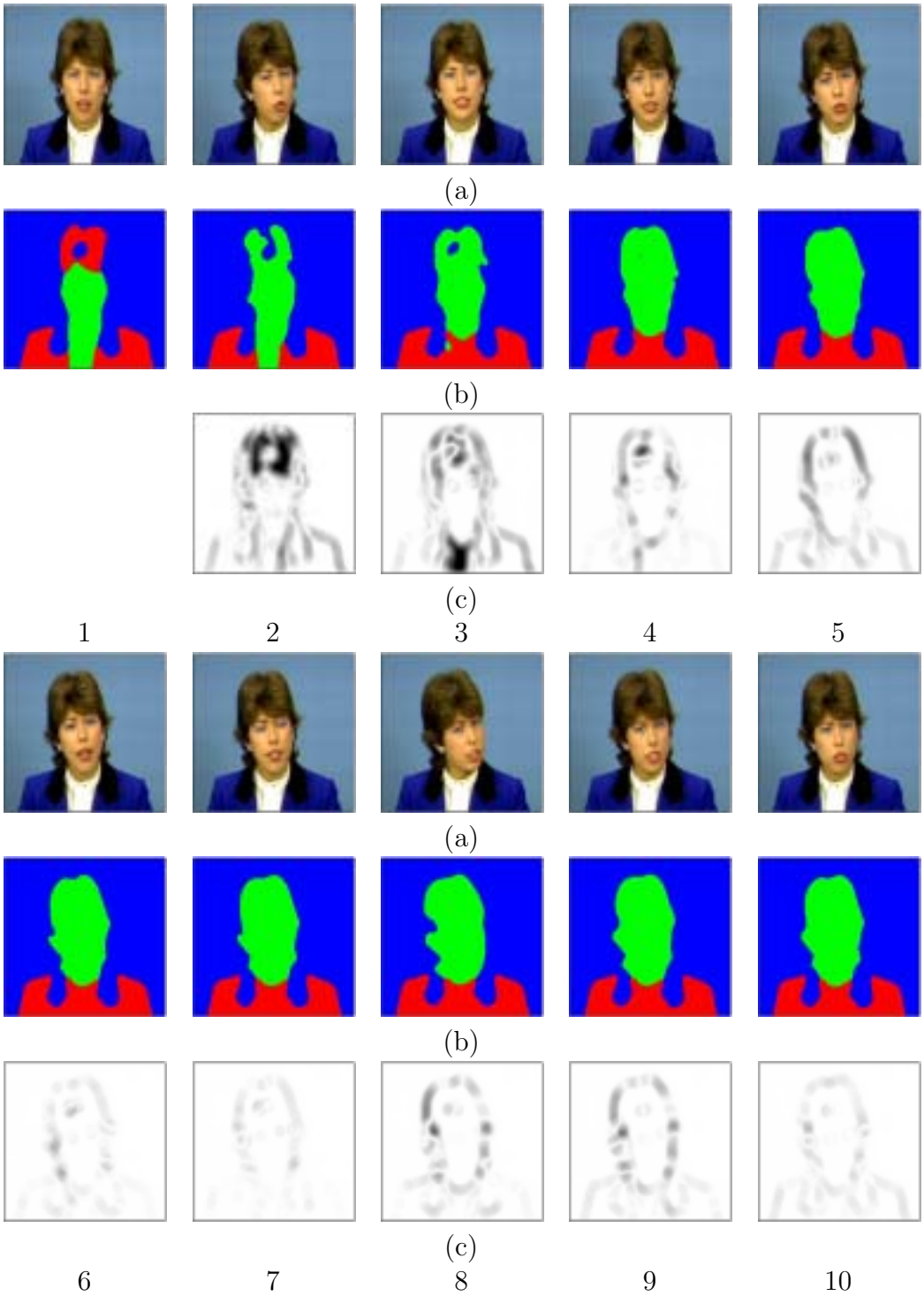


Figure 4.147: Results of applying the proposed segmentation and change detection method on *Claire* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask.

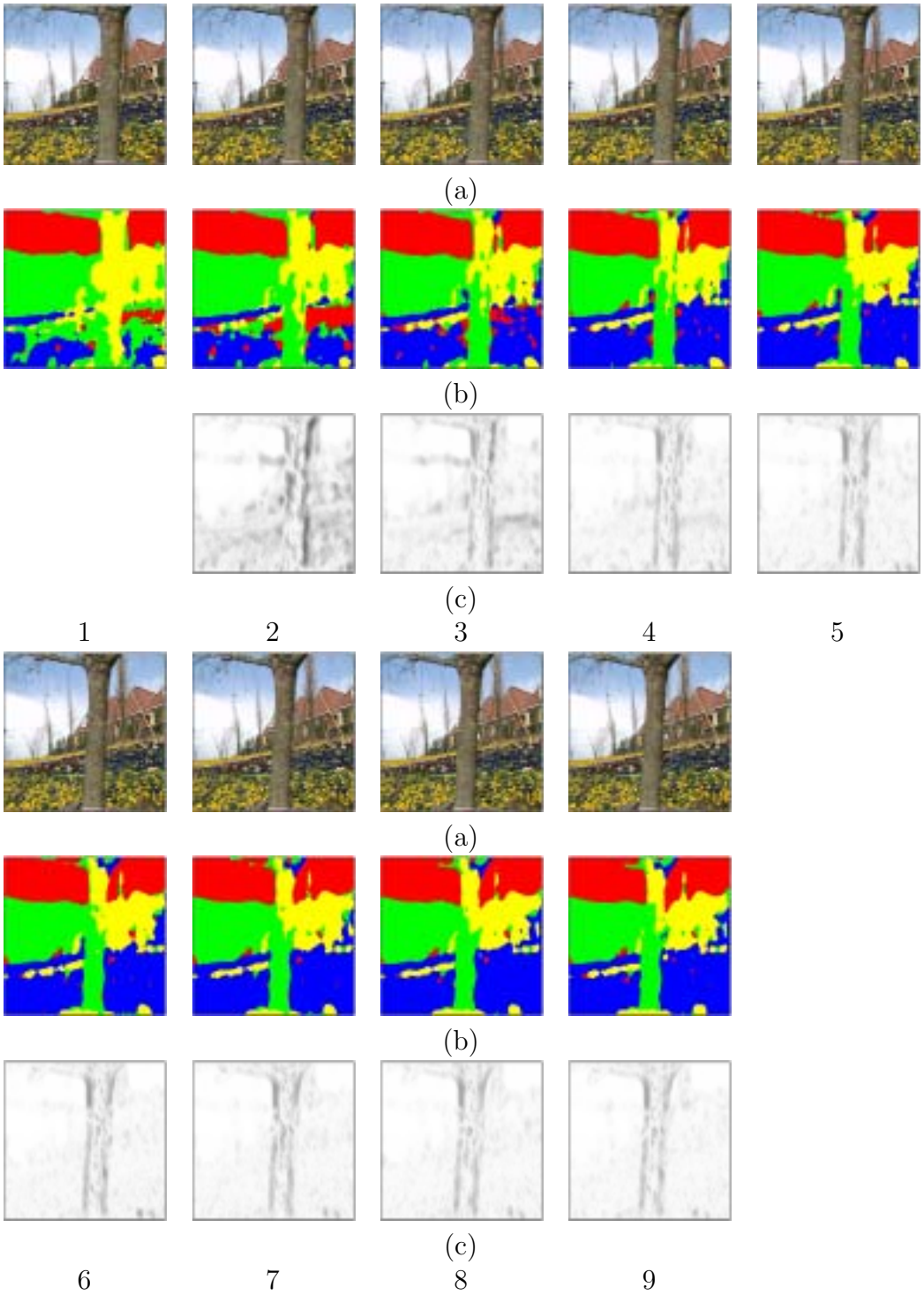


Figure 4.148: Results of applying the proposed segmentation and change detection method on *Flower Garden* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask.

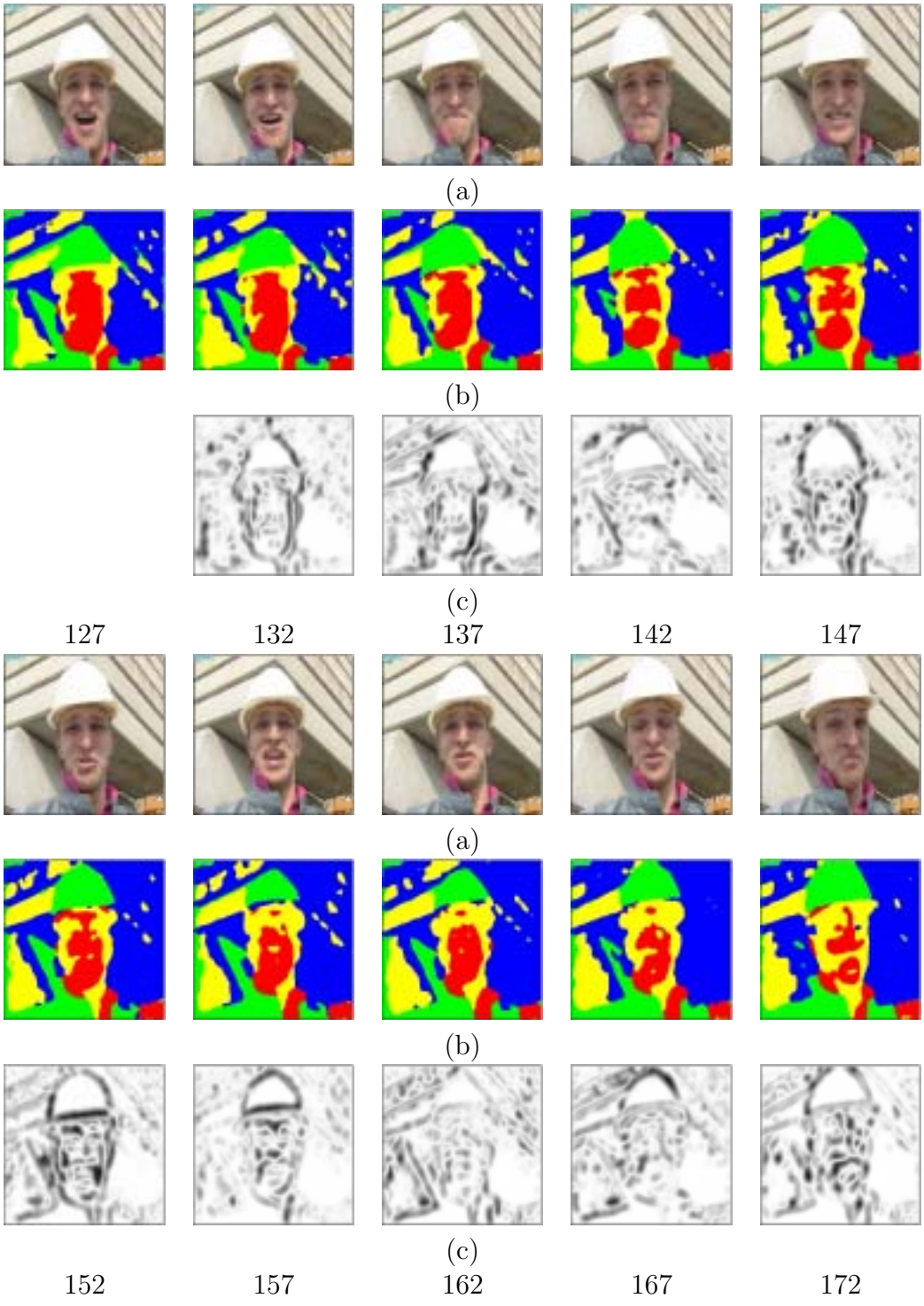


Figure 4.149: Results of applying the proposed segmentation and change detection method on *Foreman* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask.

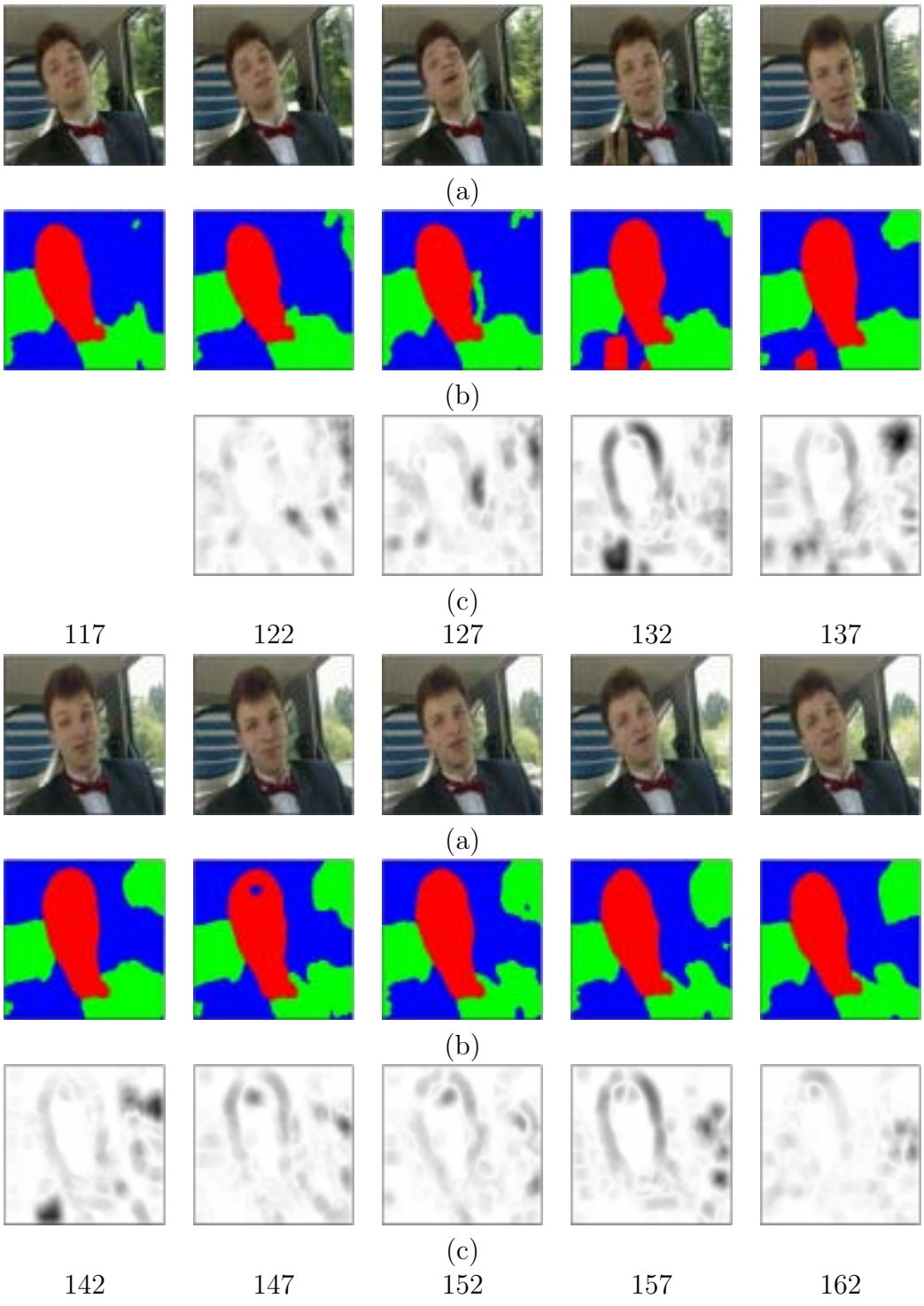


Figure 4.150: Results of applying the proposed segmentation and change detection method on *Car Phone* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask.

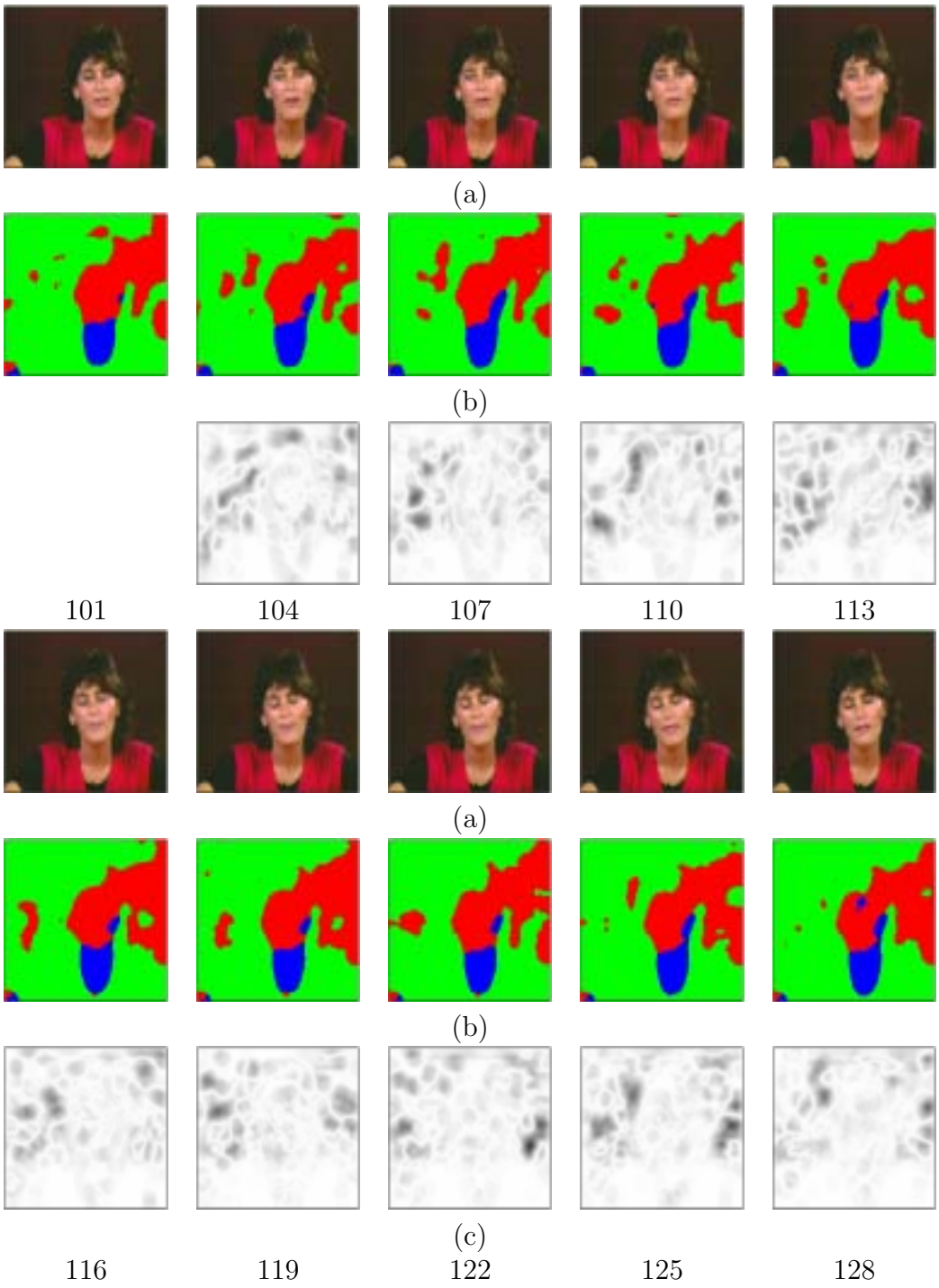


Figure 4.151: Results of applying the proposed segmentation and change detection method on *Miss America* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask.

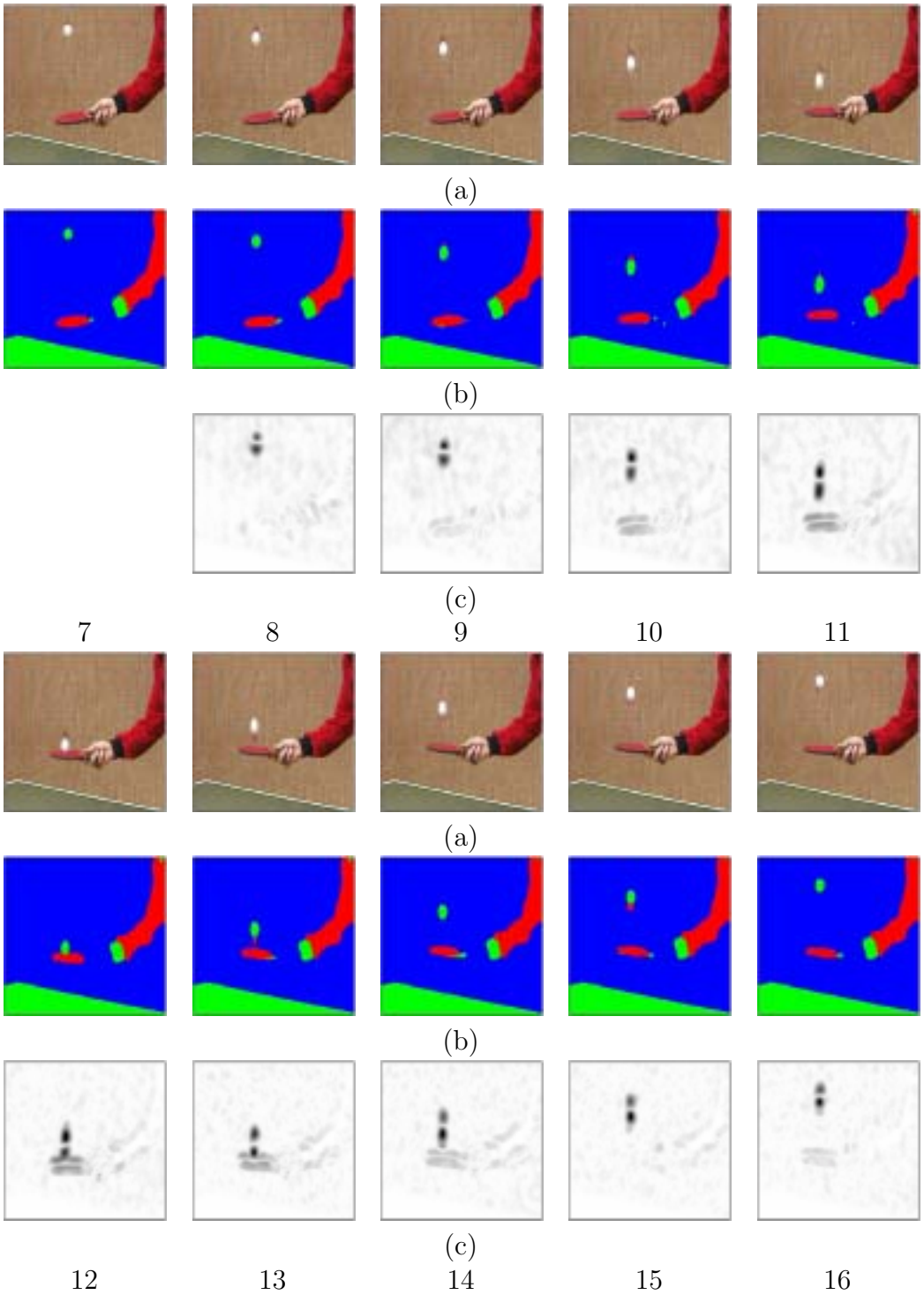


Figure 4.152: Results of applying the proposed segmentation and change detection method on *Table Tennis* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask.

41–60 interval and the scene abruptly changes in the 131–th frame. The proposed method adapts both the segmentation results and the change detection maps when scene changes gradually. Note the density of black points (changed points) while the algorithm converges after the gradual change. It has never been observed that any kind of change results in instability of the algorithm. Also, it has no important effect on the elapsed time. The algorithm also responds promisingly to the abrupt change. While the segmentation maps is getting fit to the new scene, the change map makes a high peak. After a few (about four) frames algorithm converges to the new situation. We emphasize that almost all methods relying on motion estimation do not show such robustness [362]. In those methods, extra attention should be spent on detecting scene change. When this event occurs those algorithms have to neglect all the information and totaly restart.

The proposed method computes both segmentation and fuzzy change detection at the same time. In many applications like video communication, these two primitives are used at the same time. Also, the proposed method needs no supervision. In fact, no obvious relation between the actual value of c and the results have been observed. Thus, we have selected different values of c with different samples. The value of δ is also left 0.05 in all tests. This non–parametrical performance of the proposed method is a definite benefit of the proposed method over the available literature.

Note that the time costs of the proposed method are measured when working on

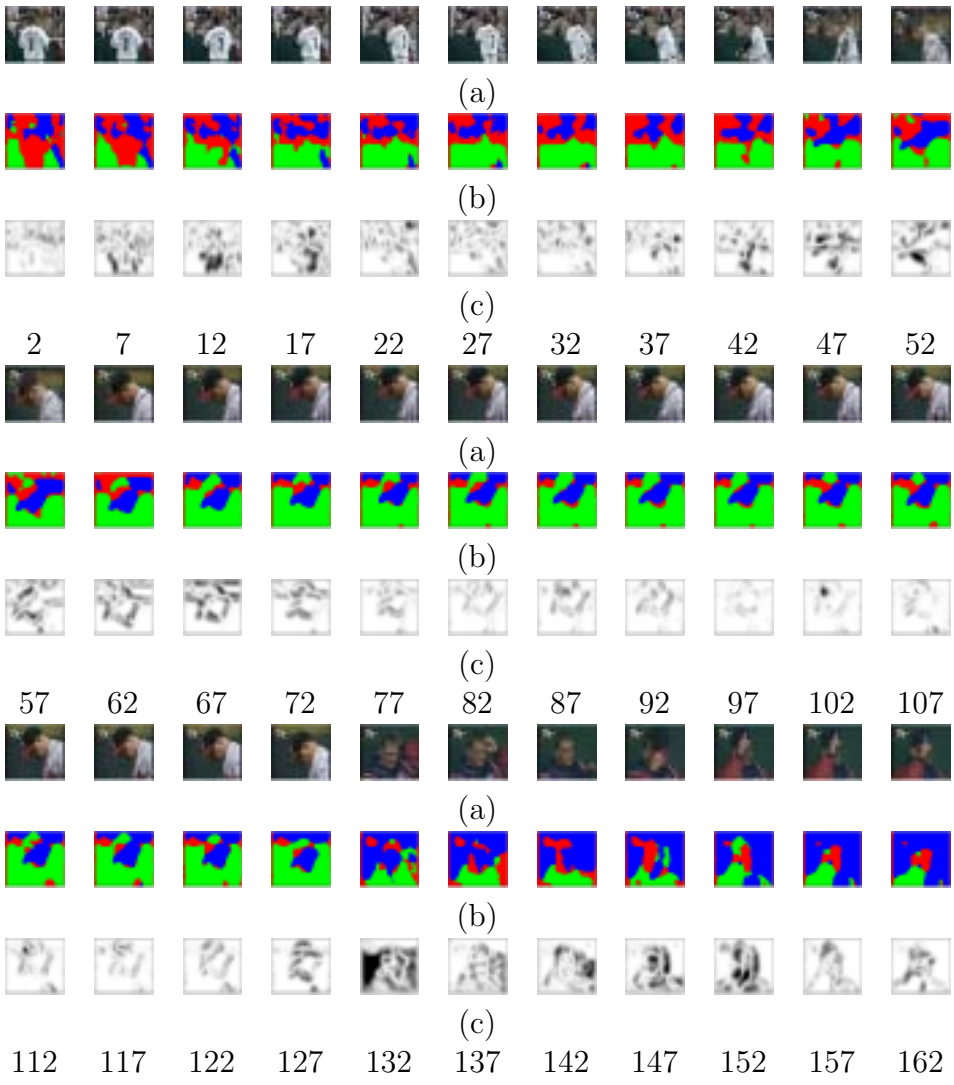


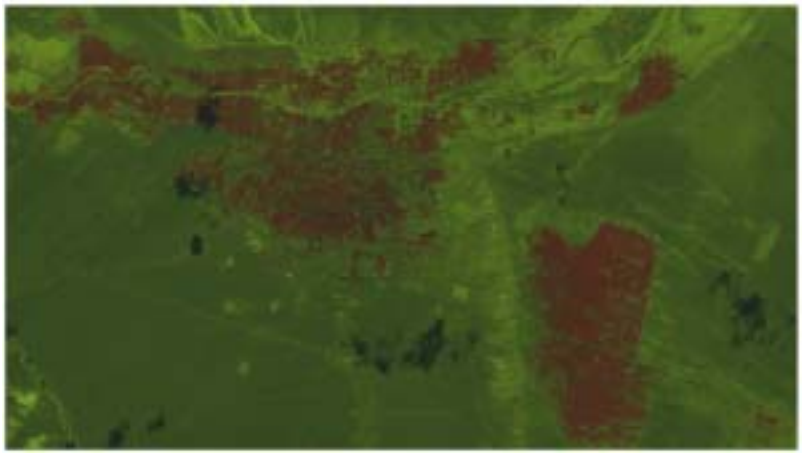
Figure 4.153: Results of applying the proposed segmentation and change detection method on *Baseball* image sequence. (a) Original frames. (b) Segmentation results. (c) Fuzzy change detection mask.

original–sized images, while many researches report the elapsed time when working on down–sampled frames. Now, performing a $1 : \lambda$ down–sampling stage on the images, results in the elapsed times scaled by $\frac{1}{\lambda}$ (many researchers use down–sampling ratio of at least $\frac{1}{4}$). A $1 : 2$ down–sampling results in a less than second elapsed time cost of the proposed method on each frame. We refer the reader to Section 2.3.1 to compare the few seconds elapsed time of the proposed method with many minutes cost of the best available video segmentation methods.

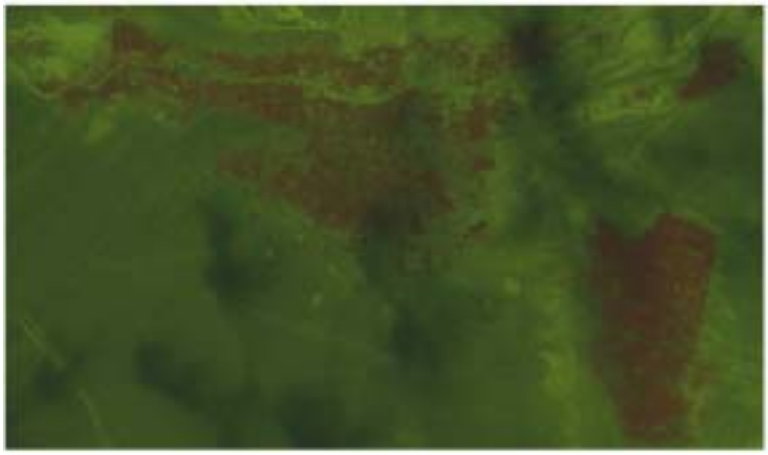
4.14 Change Detection in Remote Sensing Images

Figure 4.154 shows two images of the city of *Bam* by the *IRS–1D* satellite *LIS III* sensor, each containing 3 channels.

Figure 4.155 shows three query regions in the image shown in Figure 4.154–a, corresponding to the urban area, desert, and river bed, respectively. Figure 4.156 shows the results of segmenting the original image into three segments using the *maximum likelihood* (ML) method with the three likelihood measures of Euclidean, Mahalanobis, and LPRE. Note the cluster shapes in Figures 4.156–a, c, and e and the corresponding segmentation results in Figures 4.156–b,d, and f, respectively. As Figure 4.156 shows the Euclidean and the Mahalanobis distances have misclassified many portions of the desert into the urban class, while the LPRE shows the best compliance with the physical classes. The same results are observed in other numerous examples. Thus, the LPRE–based likelihood measure is the best of all descriptor for



(a)



(b)

Figure 4.154: Multi–spectral images of the city of *Bam*. (a) 2003-12-04. (b) 2003-12-29.

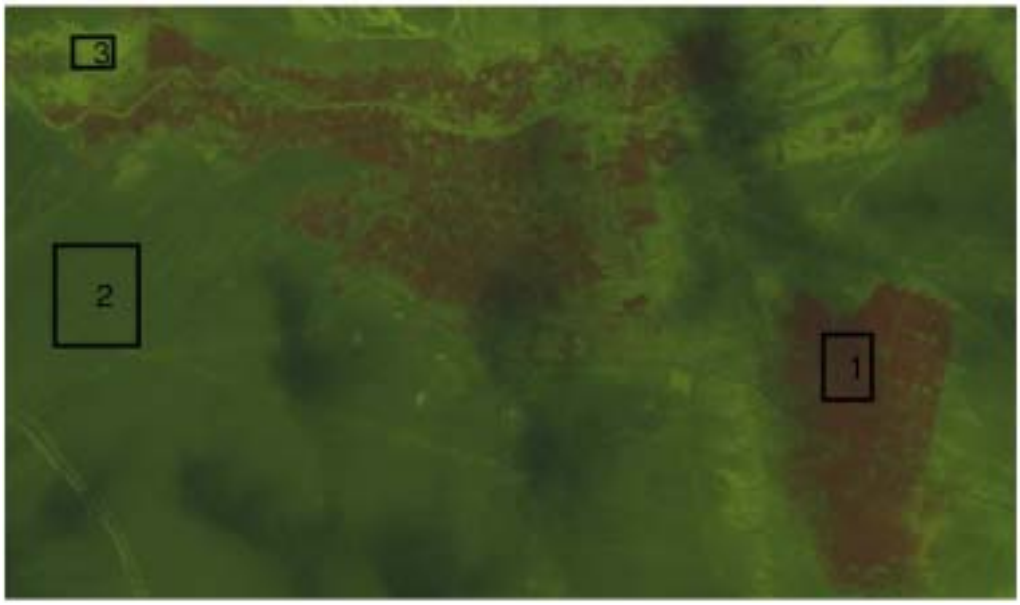
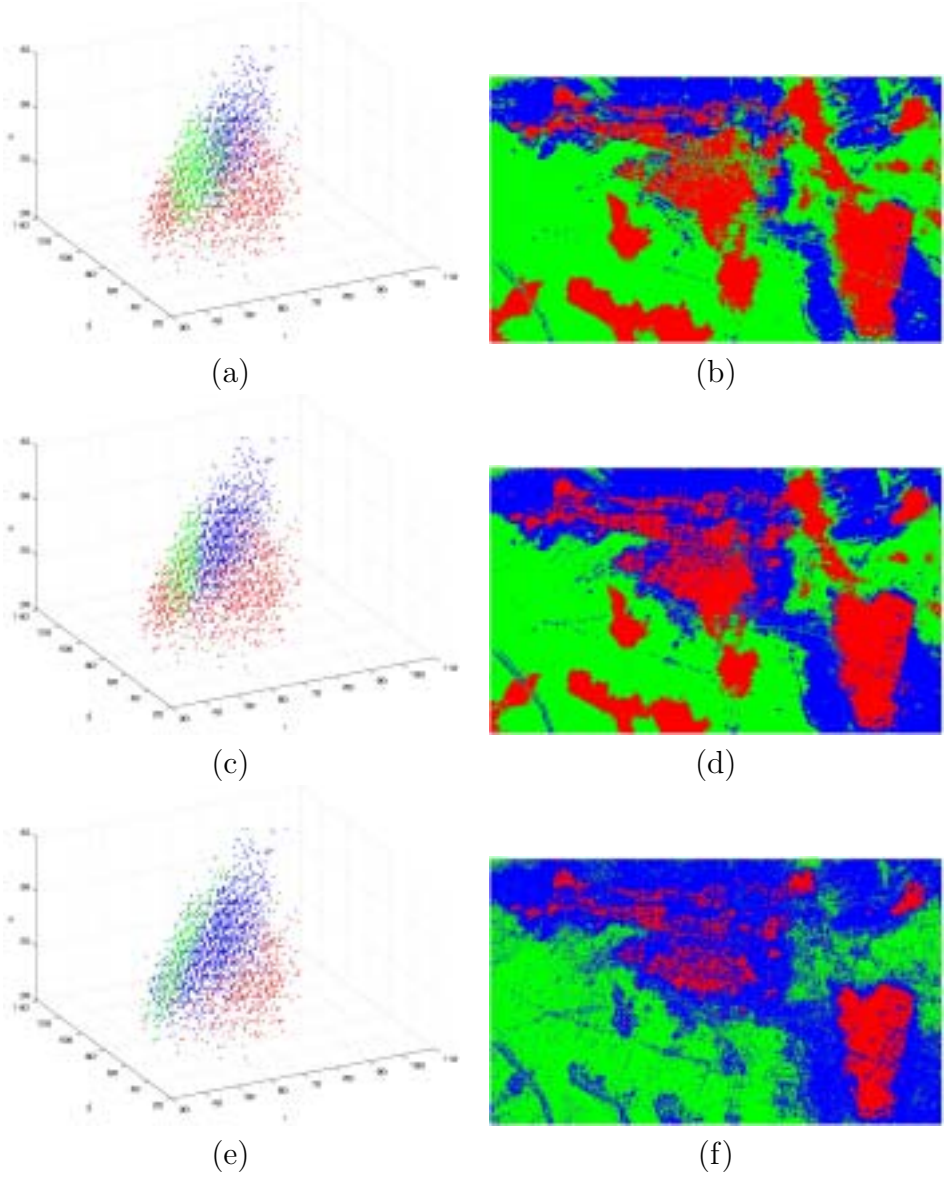


Figure 4.155: Three query regions in the image shown in Figure 4.154–a.

the images under investigation. Specially note that unless for the LPRE, the two other measures have failed to classify the non–urban areas in numerous situations and have included them in the urban class. This is so important, because after the earthquake, some spectral vectors in the urban class are getting members of the non–urban class, due to the destruction.

Figure 4.157 shows the result of applying the proposed multispectral segmentation method on the image shown in Figure 4.154–a, with the parameters set as, $c = 3$, $m = 1\frac{1}{3}$, $\delta_T = \frac{1}{2}$, and $k = 2$.

Figure 4.158 shows the result of the proposed multi–spectral fuzzy change detection applied on the image shown in Figure 4.154 with the parameters set as $c = 3$,



(a) (b)

(c) (d)

(e) (f)

Figure 4.156: Results of applying maximum likelihood on the query regions shown in Figure 4.155. (a) Euclidean. (b)Mahalanobis. (c) LPRE.

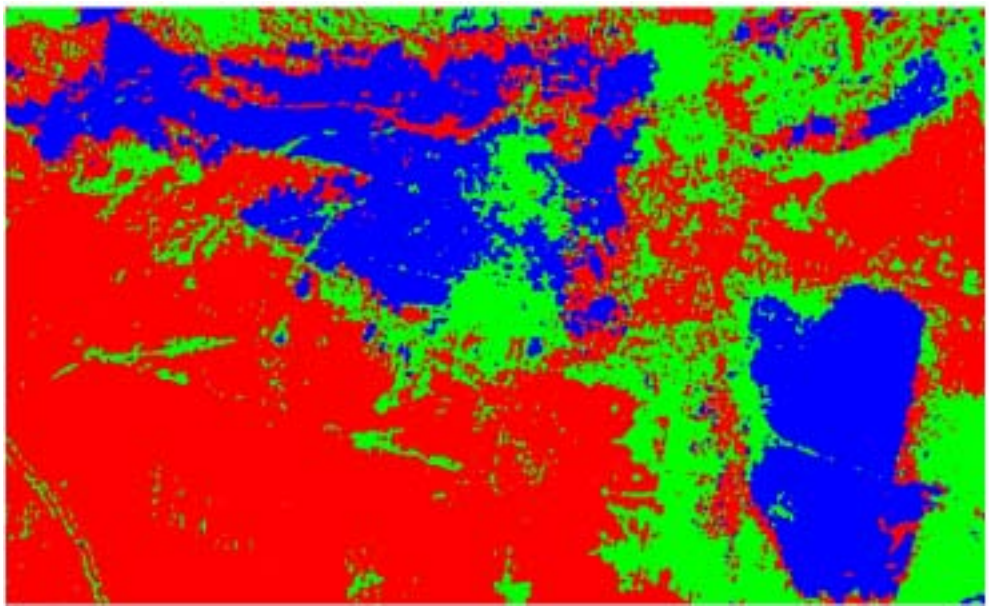


Figure 4.157: Result of the proposed segmentation method applied on the image shown in Figure 4.154–a.

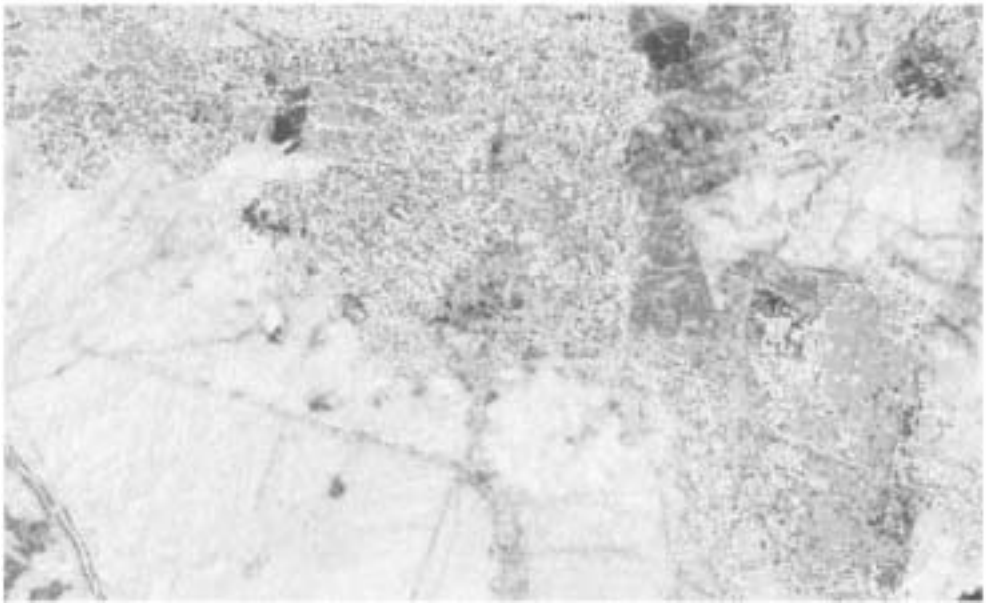


Figure 4.158: Result of the proposed multi–spectral fuzzy change detection method applied on the images shown in Figure 4.154.

$m = 1\frac{1}{3}$, $\delta_T = \frac{1}{2}$, and $k = 2$. To get a better printing result, the values of $1 - \delta_{xy}$ are shown here. Note the rightmost–bottom part of the city, were destructed parts are visible, and the corresponding black spots in the change map. Also, note that the desert part of the image is ranked to be less changed than the urban part, as desired. The main drawback of the proposed method is its sensitivity to the clouds. We propose using the channels which are less sensitive to the spectrum of the clouds.

Table 4.36 compares the proposed method with the available literature. The comparison is made in terms of the dimension of the input data (multispectral vs.

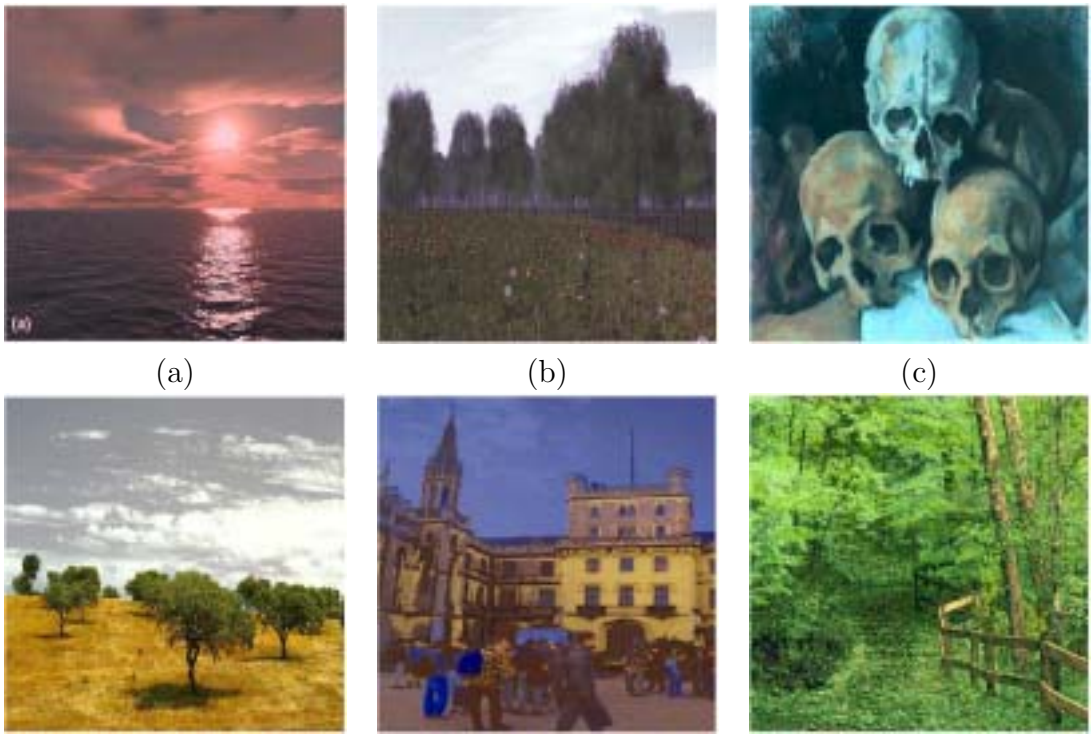
Table 4.36: Comparison of the proposed method with the literature. [MS: Multi–spectral, NM: Non Model–based, U: Unsupervised]

Method	**MS**	**NM**	**Local**	**Spatial**	**U**	**Fuzzy**
[386]	√	√	–	√	–	–
[59]	–	√	–	–	–	–
[265]	–	–	–	√	–	–
[260]	–	√	–	–	–	–
[76]	–	√	–	–	–	–
[111]	–	–	–	–	–	–
[82]	√	√	–	√	–	–
[379]	√	√	–	–	√	–
[293]	–	√	√	√	–	–
[260]	√	–	–	–	–	–
[51]	–	–	–	–	–	–
Proposed	√	√	√	√	√	√

grayscale), model–based tendency of the method, locality of change detection (adaptability to different effects of the change in different spectral bands), spatial information inclusion, human intervention (supervised vs. unsupervised), and the output space (crisp vs. fuzzy). Model–based methods are those using the prior knowledge about the image, such as information about the scale, contents, material and so on. As shown in Table 4.36, while the proposed method is the only fuzzy–output one, it works unsupervised and processes the changes locally with no need to prior information about the image contents.

4.15 Image Sequence Recoloring

Here, we first compare different recoloring methods in terms of the quality of the results. Figure 3.115 shows some of the sample images used for this comparison.



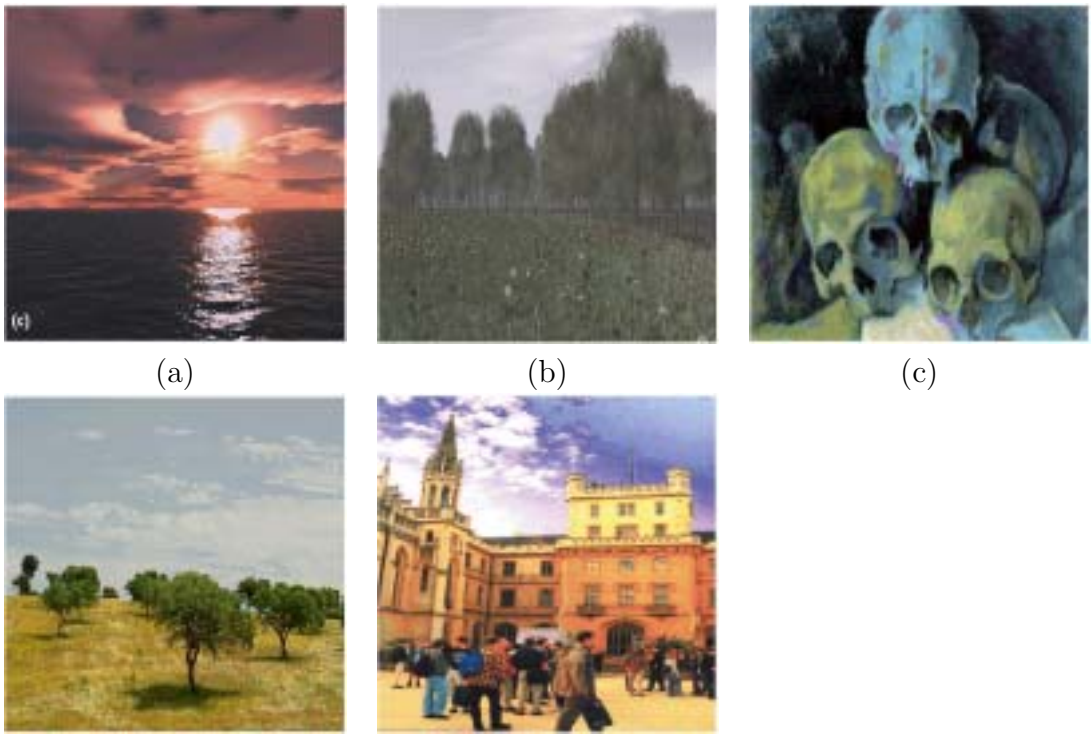
(a) (b) (c)

(d) (e) (f)

Figure 4.159: Results of the proposed color transfer method. See Table 4.37 for details.

The proposed recoloring method is performed on the sample images (illustrated in Figure 4.107). The results of the proposed method and the results of other methods are shown in Figures 4.159 and 4.160, respectively. Here, the method discussed in Section 3.9.2 is not analyzed because its results are almost identical to the proposed method.

Comparing the results shown in Figures 4.159 and 4.160 shows the superiority of the performance of the proposed method. Note that, while there is a discontinuity in



(a) (b) (c)

(d) (e)

Figure 4.160: Results of other available recoloring methods. (a),(b),(e) Method in [297]. (c) Method in [156], (d) Method in [92]. For details see Table 4.37

Table 4.37: Source and reference images corresponding to the destination images shown in Figure 4.159 and Figure 4.160.

Destination	**Source**	**Reference**
Figure 4.159–a	Figure 4.107–a	Figure 4.107–e
Figure 4.159–b	Figure 4.107–b	Figure 4.107–f
Figure 4.159–c	Figure 4.107–c	Figure 4.107–g
Figure 4.159–d	Figure 4.107–d	Figure 4.107–h
Figure 4.159–e	Figure 4.107–i	Figure 4.107–j
Figure 4.159–f	Figure 4.107–k	Figure 4.107–l
Figure 4.160–a	Figure 4.107–a	Figure 4.107–e
Figure 4.160–b	Figure 4.107–b	Figure 4.107–f
Figure 4.160–c	Figure 4.107–c	Figure 4.107–g
Figure 4.160–d	Figure 4.107–d	Figure 4.107–h
Figure 4.160–e	Figure 4.107–i	Figure 4.107–j

the sky in Figure 4.160–e, the sky in Figure 4.159–e simulates a real night sky (note that the photograph is taken at daytime). This fact is more visible when one confirms that the recolored image in Figure 4.160–e does not pretend to be a night scene while Figure 4.159–e does so. This is also visible in Figure 4.160–b where the blue sky has nothing to do with the entirely white sky of Figure 4.107–f. The sky in Figure 4.159–b is more whitened than Figure 4.160–b (as desired). Although, the method in [156] has tried to transfer the cold colors of Figure 4.107–g to Figure 4.107–c, Figure 4.160–c still contains the warm violet–blue color on the topmost skull. In Figure 4.159–c, produced by the proposed method, the colors are mostly cold. Figure 4.160–d made by the method in [92] seems acceptable in spite of the fact that the method that has produced it, is a massively expensive operation.

Thus, while the proposed method and the method introduced in Section 3.9.2 give

proper results, the performance of the methods in [297, 156] are non–satisfactory. The results of the method in [92] are also acceptable. The reason behind the poor results of the method in [297] is the assumption about the existence of a unique PCA matrix. Although, this idea has enhanced the computational cost of the method in [297] extensively (see Section 3.13.2), it has declined the quality of its output. It must be emphasized that the lack of user interaction in the methods in [92, 156] is greatly disadvantageous. In fact, any recoloring method should work on user suggestions. The method in [156] tries to fill this gap by employing suspicious pallet–assignment clues like comparing the occupied areas by pallet representatives. The approach is expressed to be problematic [156]. The method in [92] limits the color transfer between color vectors in the same fixed category and thus fails to produce desirable results such as the ones produced by the method in [297] and Section 3.9.2 and the proposed method (see Figures 4.160–a and 4.159–a).

To compare different recoloring algorithms in terms of realtime application we assume working on 512×512 image in a 25 frames per second framework with 50×50 swatches. Also, we consider implementing the algorithms on DSP processors like $DIOPSIS^{TM}740$ by *Atmel* [12] or $TMS320C6713$ by *Texas Instruments* [13] (came into market in 2001) which give the ability to process in one Giga flops per second. For a particular method which elapses a certain amount of flops on a single image, we define the utilization time as the 10–th base logarithm of total seconds needed

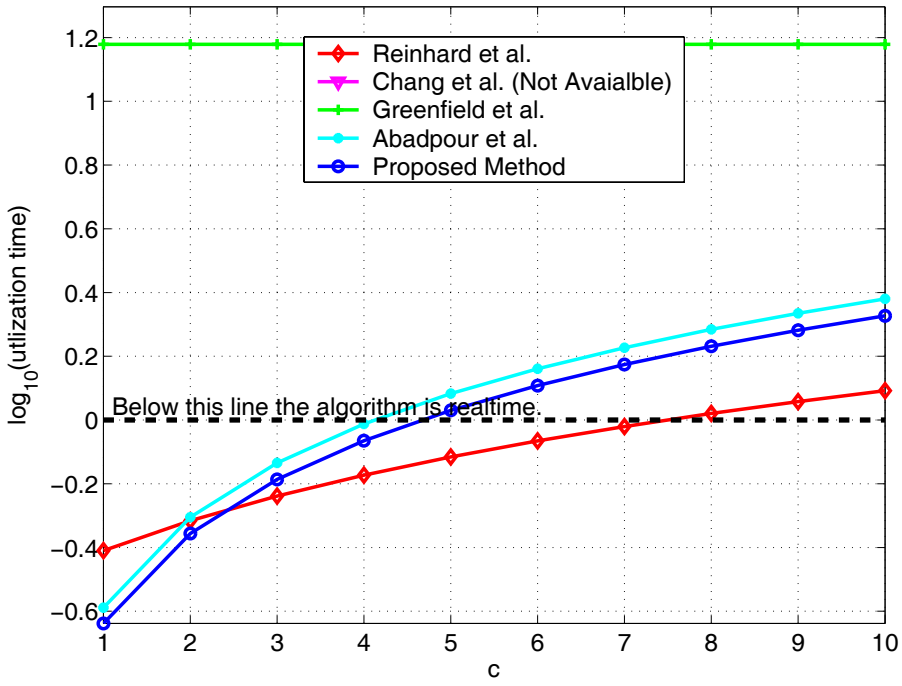


Figure 4.161: Utilization time of different methods in a 1Gflop processor.

to accomplish the job on one second of data flow. Figure 4.161 illustrates the utilization time of different algorithms for different values of c on a 1Gflop processor. In this chart, points below zero ($log_{10}1$) show realtime applications. Investigating Figure 4.161 shows that the methods in [156, 92] are definitely non–applicable for realtime applications. On the other hand, the methods in [297] and Section 3.9.2 and the proposed method are conditionally realtime. In fact, for less than seven pairs of swatches, the method in [297] is realtime while the method in Section 3.9.2 and the proposed method only allow the realtime application of three and four pairs of swatches, respectively.

Here, we propose a new hardware architecture for realtime utilization of the proposed recoloring method for image sequences. We propose slice the image into four non–overlapping images and to employ four separate DSP processors. Each processors receives all pairs of swatches along with its portion of the image. According to the fact that the method in [297] and Section 3.9.2 and the proposed method work on image pixels independently, putting back the four parts of the image in their recolored format gives the recolored version of the whole image. The overhead of the proposed hardware implementation contains data transfer between processors (only before the process begins and after it finishes) and the excessive cost of computing PCA for swatches four times, repeatedly. As this cost is in order of swatch size (cwh), and not the image size (CWH), we neglect it and assume a 5% excess in computation for it. Figure 4.162 illustrates the proposed architecture and Figure 4.163 shows the results of simulating the proposed method on the proposed architecture. Note that in this analysis, we have not considered the parallel structure of the proposed method which drastically enhances its performance.

Figure 4.164 compares the utilization time of different methods in the new architecture. Note that the methods in [156, 92] are not accomplishable in this new architecture. The methods in [297], the one in Section 3.9.2, and the proposed method are working in realtime in this architecture. Table 4.38 compares different recoloring method in terms of computational cost, user contribution, and quality of results.

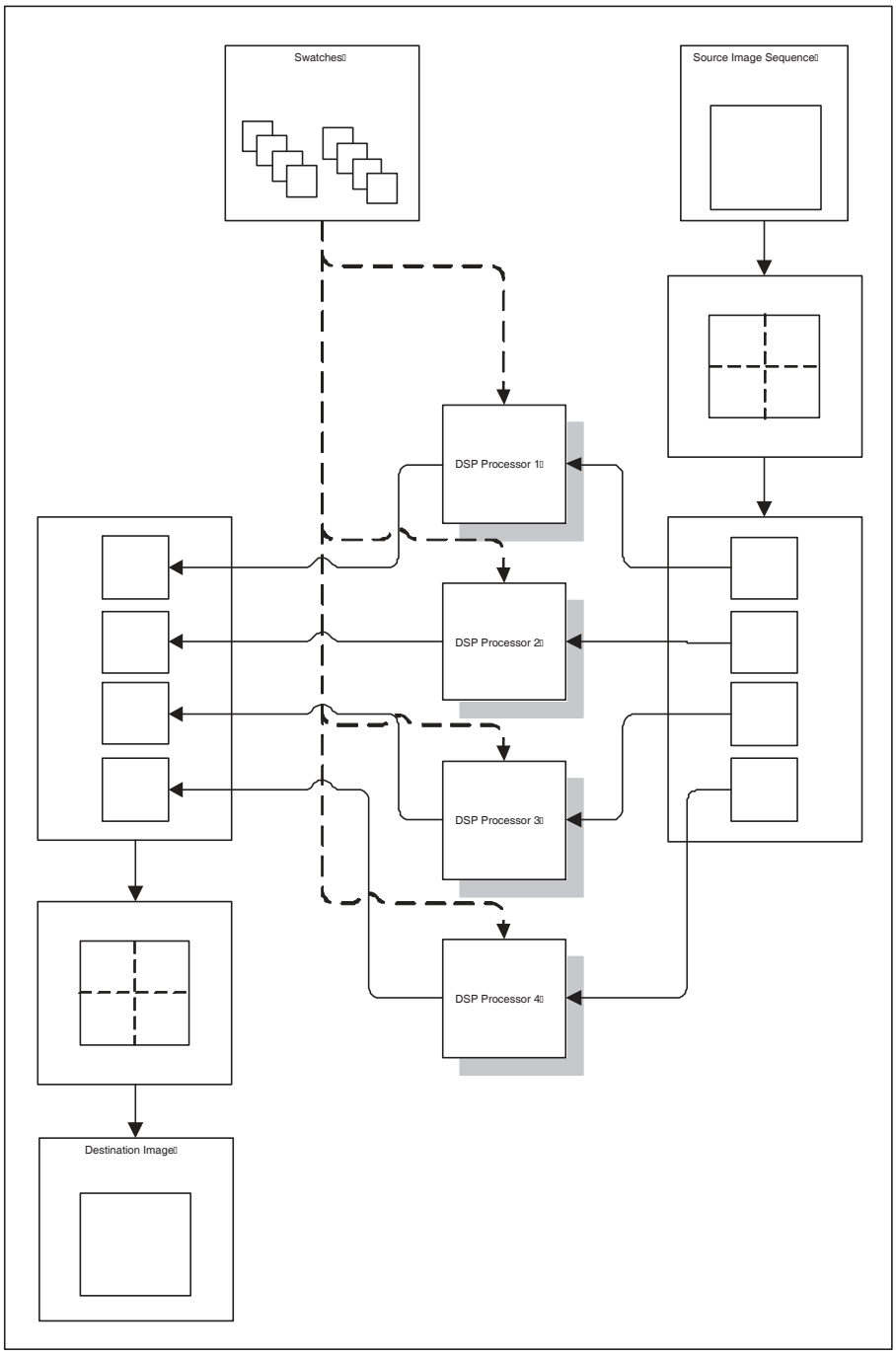


Figure 4.162: Proposed four–processor architecture used for the realtime application of the recoloring methods.

(a) (b) (c)

Figure 4.163: Simulation of the proposed method in the proposed architecture. (a) Source image. (b) Reference image. (c) Recolored image.

Table 4.38: Comparison of different recoloring method in terms of realtime application, user contribution, and quality of results. All methods are evaluated when performing on $W \times H$ color images and c pairs of watches.

Method	**Realtime Application**	**User Contribution**	**Quality of Results**
[297]	Yes	Allowed	Weak
[92]	No	Not allowed	Fair
[156]	No	Not allowed	Weak
Section 3.9.2	Yes	Allowed	Fair
Proposed Method	Yes	Allowed	Fair

As Table 4.38 shows the proposed method and the method in Section 3.9.2 are the only realtime methods that give fair results while enabling user contribution. The proposed method is about 11% faster than the method in Section 3.9.2. Also the proposed method eliminates the two non–friendly parameters of α and β.

Figures 4.165, 4.166, and 4.167 show the results of applying the proposed recoloring method on the standard image sequences of *Table Tennis*, *Clair*, and *Flower*

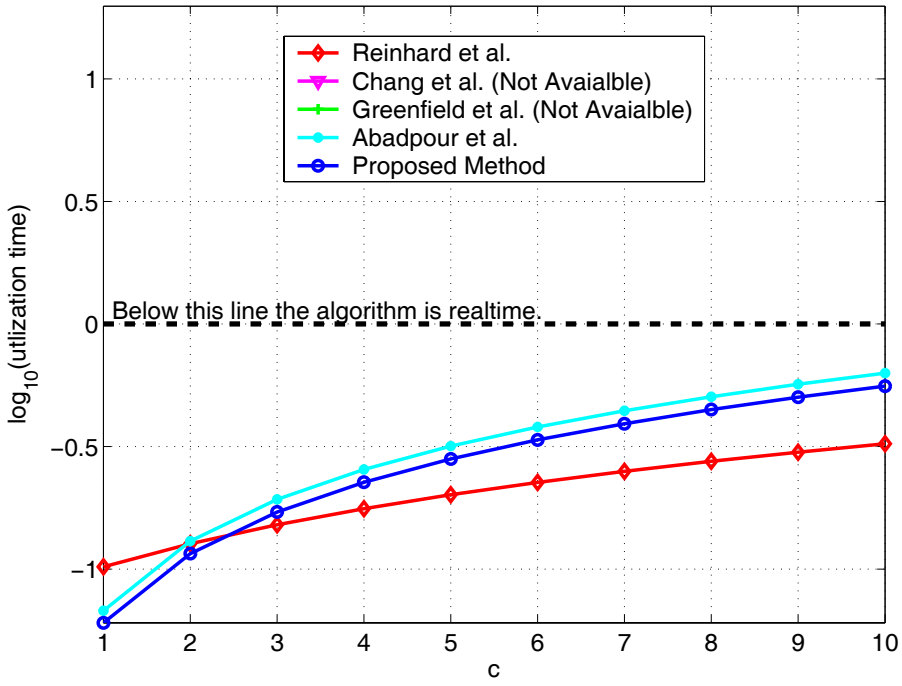


Figure 4.164: Utilization time of different methods in the proposed architecture.

Table 4.39: Numerical details about the results in Figures 4.165, 4.166, and 4.167. [m: Fuzziness, c: Number of swatches. ρ_+, ρ_-: Convolution kernels' sizes. t: Elapsed time.]

Name	**Size**	**Input**		**Parameters**		**Output**	
		Src.	**Ref.**	m	c	$t(s)$	**Result**
Table Tennis	240×352	4.168–a	4.168–b	2	4	1.38	4.165
Clair	208×240	4.168–c	4.168–d	1.1	4	0.93	4.166
Flower Garden	240×352	4.168–e	4.168–f	1.1	3	1.14	4.167

Garden. In each figure the original frames are shown in the top row, while the bottom row contains the results. Figure 4.168 shows the swatches used for these three recoloring tasks and Table 4.39 lists the numerical results and parameters. Note the quality of recolored images in Figures 4.165, 4.166, and 4.167. Also, note the acceptable time elapsed by the proposed process on each frame. These elapsed times are recorded for a non–optimized code.

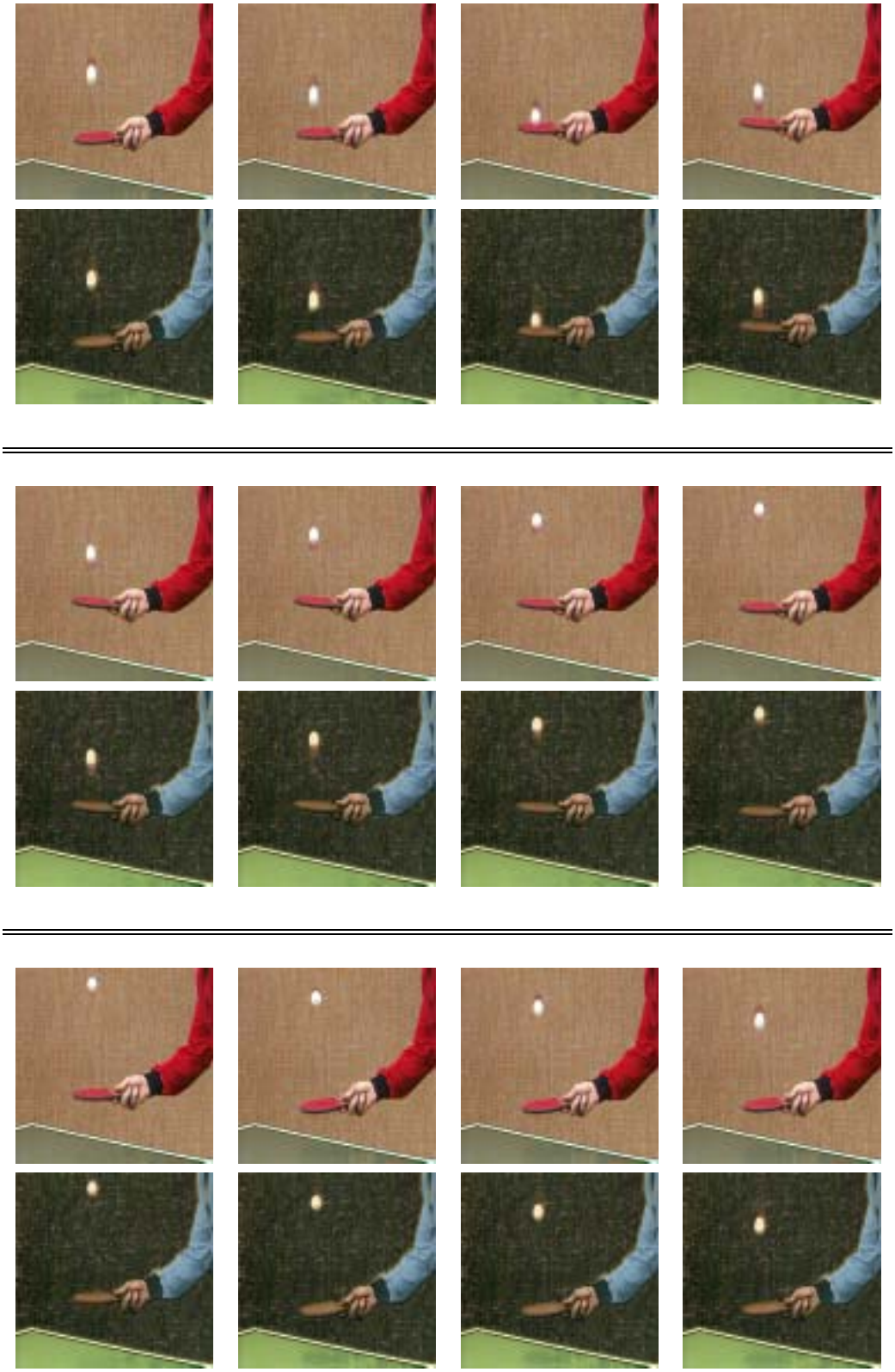
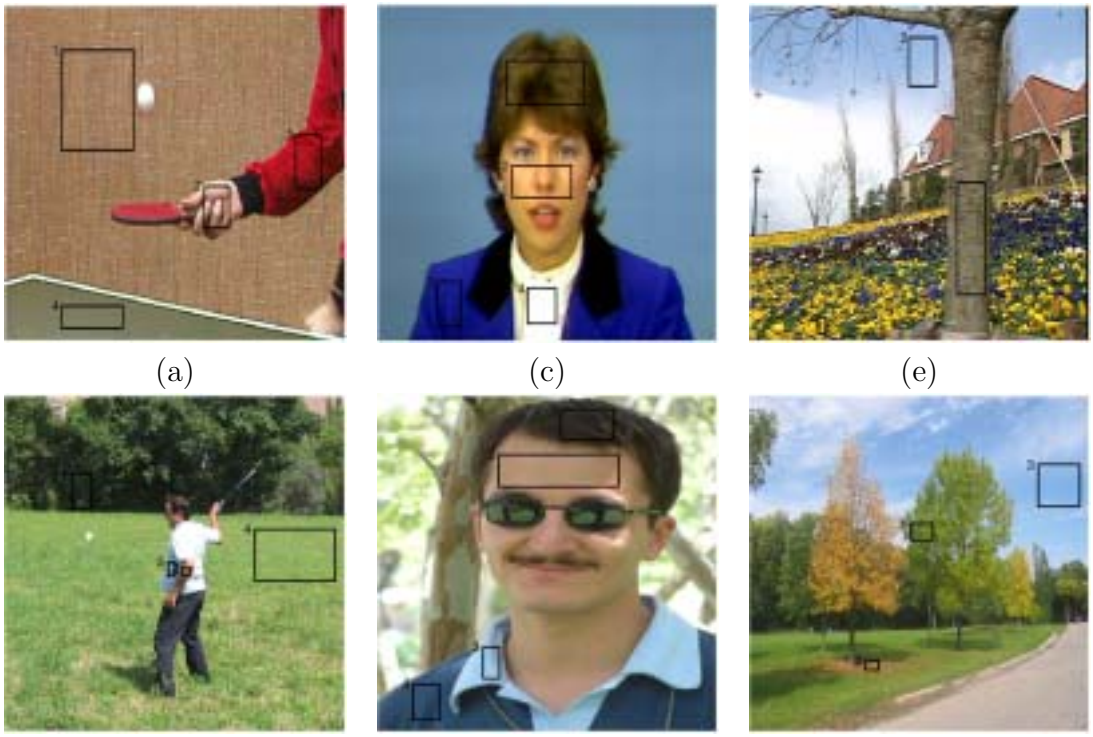


Figure 4.165: Results of applying the proposed recoloring method on the *Table Tennis* standard image sequence (for details see Table 4.39).

Figure 4.166: Results of applying the proposed recoloring method on the *Clair* standard image sequence (for details see Table 4.39).

Figure 4.167: Results of applying the proposed recoloring method on the *Flower Garden* standard image sequence. (for details see Table 4.39).



(a) (c) (e)

(b) (d) (f)

Figure 4.168: Swatches used for recoloring processes shown in Figures 4.165, 4.166, and 4.167. For details see Table 4.39.

Conclusions

Using numerous experiments, the properness of linear dimension reduction in homogenous color swatches is investigated. The results show that, in accordance to the theoretical prediction, applying principal component analysis (PCA) on color vectors of homogenous swatches of images of the nature gives acceptable 1– and 2–D representations. The existence of the extra dimension(s) plus the suitability of linear modelling enables many efficient algorithms for processing color images. In this thesis, new methods for performing spotting, colorizing, recoloring, segmentation, compression, watermarking, query region extraction, and deliberate distortion (partial delivery) are proposed and experimentally analyzed. We believe that these methods are only examples for using PCA for processing color images. The results of this thesis indicate that when trying to design an algorithm for color image processing, PCA should one of the first tools which are examined.

Appendix A

Color Image Processing (CIP) Toolbox

The code developed for this thesis is gathered in an integrated toolbox for MATLAB. As such, at the time of writing this phrases, the *color image processing* (**CIP**) toolbox revision 5.3 includes 181 functions.

The file structure of **CIP** is as follows:

- **ColorSpaces**: Color Space Conversion functions. See Section 2.1.
- **Data**: This directory contains *.mat* files (MATLAB data files) which are used by other functions. See Section A.3.
- **Methods**: Plenty of published color image processing methods. These functions are described in Chapter 3 in parallel with the mathematics of them. Some of the functions which are not described there, are demonstrated in Section A.4.
- **SamplesFew**: Three color and three grayscale sample images included in the

suite version of the toolbox.

- **SamplesMany**: All samples including 2013 color and 349 grayscale images in revision 5.3, 24 June 2005 (see Appendix B).

- **Utility**: Utility functions. See Section A.1.

- **Visualization**: Functions for visualizing special data related to color image processing. see Section A.2.

A.1 Utility functions

CIP includes many utility functions which are helpful for developing color image processing algorithms. For example, the function *RCIPIS* gives the index to the last sample image. The function inputs an argument which determines wether a color image or a grayscale image is required (See Figure A.1). the function *RCIPIS* internally uses *LCIPIS* to obtain the last index to the sample images (See Figure A.2).

CIP includes the three functions *epsifyfig*, *epsifyI*, *epsifyf* for producing EPS files from within the code. The three function convert the contents of a figure, a variable representing an image, and an image file to an EPS file, respectively. Figures A.3, A.4, A.5 shows these functions.

The function *grect* is defined in **CIP** for helping developers produce a proper user interaction when selecting a rectangular area in the image is needed. The function is

RCIPIS

```
function n=RCIPIS(color)
%Funciton randomized CIP image sample gives a random index to a CIP
%sample image.
%Usage: n=RCIPIS(color);
%Inputs: color:A color sample? (default=1)
%Outputs:
%Revision 1.0 Arash Abadpour, Spring 2005 (March 31, 2005)
```

Figure A.1: Implementation of the *RCIPIS* function.

LCIPIS

```
function n=LCIPIS(color)
%Funciton last CIP image sample gives the index of the last image sample of
%the color image processing toolbox.
%Usage: n=LCIPIS(color);
%Inputs: color:A color sample? (default=1)
%Outputs:
%Revision 1.0 Arash Abadpour, Winter 2005 (March 3, 2005)
```

Figure A.2: Implementation of the *LCIPIS* function.

epsifyfig

```
function epsifyfig(fig,s2,color)
%Function epsifyf converts figure fig into eps.
%
%Usage:epsifyfig(fig,s2,color);
%Inputs: fig: figure.
%        s2: EPS file.
%        color: Color EPS?
%Outputs:
%
%Revision 1.01 (c) Arash Abadpour Spring 2005 (March 16, 2005)
```

Figure A.3: Implementation of the *epsifyfig* function.

epsifyI

```
function epsifyI(I,s2)
%Function epsifyI converts the image I into eps.
%
%Usage:epsifyI(I,s2);
%Inputs: I: Source image.
%        s2: EPS file (will be produced automatically if missed).
%Outputs:
%
%Revision 1.0 (c) Arash Abadpour Fall 2004 (Nov. 20 2004)
```

Figure A.4: Implementation of the *epsifyI* function.

epsifyf

```
function epsifyf(s1,s2)
%Function epsifyf converts the image s1 into eps.
%
%Usage:epsifyf(s1,s2);
%Inputs: s1: Image name.
%        s2: EPS file (will be produced automatically if missed).
%Outputs:
%
%Revision 1.0 (c) Arash Abadpour Fall 2004 (Nov. 14 2004)
```

Figure A.5: Implementation of the *epsifyf* function.

similar to the *getrect* function defined in MATLAB, except for some minor differences including making the user able to perform zooming. Figure A.6 shows this function.

The function *hist3* computes the 3–D histogram of the given data (See Figure A.7). Using *hist3* one have to input the given signal (for example a color image) and the number of bins. Also, user can have the function eliminate empty bins in cost of extra processing cost. The function then gives the bins and the number of occupants in each one. The results are normalized to sum to unity.

CIP includes the two functions *I2RLE* and *RLE2I* for converting a binary image to run–length code and vice versa. See Figures A.8 and A.9.

There are four functions in **CIP** which are essentially suited for processing color images. The two functions *I2V* and *V2I* use the reshape function to convert a $H \times W \times 3$ array to $3 \times WH$ representation and vice versa. This representation

grect

```
function [y1,x1,y2,x2]=grect(zooming)
%Function GetRectangle gets coordinates of a rectangle in the current
%figure, using ginput function and sorting the results.
%Usage: [y1,x1,y2,x2]=grect(zooming);
%Inputs:     zooming:Allow zooming?
%Outputs: y1,x1,y2,x2:Coordinates of a rectangle at x1:x2,y1:y2
%Revision 2.111 Arash Abadpour fall 2004 (October 2, 2004)
```

Figure A.6: Implementation of the *grect* function.

hist3

```
function [N,x]=hist3(I,bin,e)
%function three dimensional histogram computes the 3D histogram of the
%image I.
%Usage:      [N,x]=hist3(I,bin,e);
%Inputs:     I:Original image.
%           bin:Number of bins on each dimension.
%Outputs:    N: Histogram.
%           x: Bin Centers.
%           e:Perform elimination.
%Revision 1.01 Arash Abadpour Winter 2004 (Feb. 6, 2005)
```

Figure A.7: Implementation of the *hist3* function.

I2RLE

```
function S=I2RLE(I)
%Function image to run length computes the runlength code of the image I.
%Usage:      S=I2RLE(I)
%Inputs:     I:Original image.
%Outputs:    S:RLE code.
%Revision 1.0 Arash Abadpour Fall 2004 (October 2, 2004)
```

Figure A.8: Implementation of the *I2RLE* function.

RLE2I

```
function I=RLE2I(S)
%Function run length code to image computes the image version of the
%specified run length code.
%Usage:      I=RLE2I(S)
%Inputs:     S:RLE code.
%Outputs:    I:The image.
%Revision 1.0 Arash Abadpour Fall 2004 (October 2, 2004)
```

Figure A.9: Implementation of the *RLE2I* function.

I2V

```
function [V,w,h]=I2v(I)
%Image to Vector converts a w*h*3 image of type uint8 to a 3*wh vector of
%type double. V2I does the inverse.
%Usage: [V,w,h]=I2V(I);
%Inputs:  I: Image
%Outputs: V: vector
%         w: width of I
%         h: height of I
%Arash Abadpour Revision 3.0 Spring 2005 (March 25, 2005).
```

Figure A.10: Implementation of the *I2V* function.

is ideal for performing linear algebra operations like applying the PCA transform. Figures A.10 and A.11 show this two functions. The two other functions split a color image to its color components (*splitrgb*) and merge three color components to produce a single color image (*mergergb*). See Figures A.12 and A.13 for further information.

The function *icor* computes the correlation of two given images. Here, $c^f_{I_1,I_2}$, the correlation of I_1 and I_2 is defined as,

$$c^f_{I_1,I_2} = \begin{cases} \frac{E\{I_1 I_2\}}{\min\left\{\|I_1\|,\|I_2\|\right\}}, f = 0 \\ 2\frac{E\{I_1 I_2\}}{E\{I_1^2\}+E\{I_2^2\}}, f = -1 \\ \frac{E\{I_1 I_2\}}{\min\left\{E\{I_1^2\},E\{I_2^2\}\right\}}, f = 1 \end{cases} . \tag{A.1.1}$$

Here, $\|I\|$ is the portion of I occupied by non–zero members. See Figure A.14 for detailed information.

The two function *jpeg* and *pcx* are developed to ease image conversion in **CIP**. See Figures A.15 and A.16. These two function use internal MATLAB commands.

V2I

```
function [I]=V2I(V,w,h,convert_uint8)
%Vector to Image converts a a 3*wh vector of type double to a w*h*3 image of
%type uint8. I2V does the inverse.
%Usage: [I]=V2I(V,w,h,convert_uint8)
%Inputs:   V: vector
%          w: width of I
%          h: height of I
%          convert_uint8:Convert to uint8?
%Outputs: I: Image
%Revision 2.1 (c) Arash Abadpour Summer 1383 (1383/5/17)
```

Figure A.11: Implementation of the *V2I* function.

splitrgb

```
function [R,G,B]=splitrgb(I)
%Function split rgb splits the image I into three channels of Red, Green
%and Blue.
%Usage: [R,G,B]=splitrgb(I)
%Inputs:          I:Original image
%Outputs:         R,G,B:Respective channels.
%Arash Abadpour Revision 3.0 Spring 2005 (March 25, 2005)
```

Figure A.12: Implementation of the *splitrgb* function.

mergergb

```
function I=mergergb(R,G,B)
%Function merge rgb merges three channels of Red, Green, Blue into a unique
%image.
%Usage: I=mergergb(R,G,B)
%Inputs:        R,G,B:Respective channels.
%Outputs:       I:Resulting image
%Arash Abadpour Revision 2.0 Spring 2005 (March 25, 2005)
```

Figure A.13: Implementation of the *mergergb* function.

ICor

```
function x=ICor(I,J,fuzzy)
%function image correlation computes the values of 2I1*I2/(I1∧2+I2∧2)
%Usage:     x=ICor(I,J);
%Inputs:        I,J:Inputs images
%              fuzzy:0,1.
%Output:        x:Correlation
%Revision 2.3 Arash Abadpour Winter 2004
```

Figure A.14: Implementation of the *ICor* function.

jpeg

```
function [J,sz,t]=jpeg(I,quality)
%Function JPEG gives the JPEG compressed version of I.
%Usage: [J,sz,t]=jpeg(I,quality);
%Inputs:     I:Original Image.
%           Quality: 0:100.
%Ouput:      J:Result.
%           sz:File Size.
%           t:Elapsed time.
%Revision 1.12 Arash Abadpour Winter 2005 (Feb. 6, 2005)
```

Figure A.15: Implementation of the *jpeg* function.

pcx

```
function [sz,t]=pcx(I,n)
%Function PCX performs PCX compression on I.
%Usage: [sz,t]=pcx(I,n);
%Inputs:     I:Original Image.
%           n:Number of colors.
%Ouput:      sz:File Size.
%           t:Elapsed time.
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 6, 2005)
```

Figure A.16: Implementation of the *pcx* function.

pdf1d

```
function H=pdf1d(I,step,bin,i1,M)
%function one dimensional pdf computes the 1-D pdf of I with mask M.
%Usage:      H=pdf1d(I,step,bin,i1,M)
%Inputs:     I:Original image.
%                step:Image sampling ratio.
%                 bin:Number of bins.
%                  i1:Index of the channel.
%                M:Mask.
%Outputs: H:The histogram.
%Revision 1.01 Arash Abadpour Summer 2004 (1383/5/13)
```

Figure A.17: Implementation of the *pdf1d* function.

CIP includes the three functions *pdf1d*, *pdf2d*, and *pdf3d* for computing the probability distribution function of 1–, 2–, and 3–D random variables. As shown in figures A.17, A.18, and A.19, these functions are suited for computing the PDF of the subspaces of color spaces. All three functions input a color image, a step value (for down–sampling), the number of bins and the index to the channel included in the histogram. As such, *pdf1d* gets *i1* and *pdf2d* gets *i1* and *i2*. The user may give a binary mask to these functions to denote those pixels which should be included in the process. Also, **CIP** includes the function *pdfnd* which is only an interface to the three above functions (see Figure A.20).

There are four functions devoted to normalization of random variables in **CIP**. These functions are applicable for preparing a 2–D array for better visual inspection.

pdf2d

```
function H=pdf2d(I,step,bin,i1,i2,M)
%function two dimensional pdf computes the 2-D pdf of I with mask M.
%Usage:      H=pdf2d(I,step,bin,i1,i2,M);
%Inputs:     I:Original image.
%                step:Image sampling ratio.
%                 bin:Number of bins.
%                  i1,i2:Index of the channels.
%                M:Mask.
%Outputs: H:The histogram.
%Revision 1.0 Arash Abadpour Summer 2004 (1383/5/1)
```

Figure A.18: Implementation of the *pdf2d* function.

pdf3d

```
function H=pdf3d(I,step,bin,M)
%function three dimensional pdf computes the 3-D pdf of I with mask M.
%Usage:      H=pdf3d(I,step,bin,M);
%Inputs:     I:Original image.
%                step:Image sampling ratio.
%                 bin:Number of bins.
%                M:Mask.
%Outputs: H:The histogram.
%Revision 1.0 Arash Abadpour Summer 2004 (1383/5/1)
```

Figure A.19: Implementation of the *pdf3d* function.

pdfnd

```
function H=pdfnd(I,step,bin,i,M)
%function n dimensional pdf computes the n-D pdf of I with mask M.
%Usage:      H=pdfnd(I,step,bin,i,M)
%Inputs:     I:Original image.
%           step:Image sampling ratio.
%           bin:Number of bins.
%           i:Index of the channel (123, 12, 23, 31, 1, 2, and 3).
%           M:Mask.
%Outputs: H:The histogram.
%Arash Abadpour Revision 1.0 Spring 2005 (March 25, 2005).
```

Figure A.20: Implementation of the *pdfnd* function.

Though, they have been used perfectly in eigenimage–based compression. The function *normalize* shown in Figure A.21 applies the below transformation on the given random variable,

$$\tilde{x} = 255\frac{x - \eta_x}{2\sigma_x}. \tag{A.1.2}$$

Here, η_x and σ_x denote the expectation and standard deviation of the input random variable, respectively. function *normalize255* spans the dynamic range of the given random variable into the $[0, 255]$ interval (see Figure A.22),

$$\tilde{x} = 255\frac{x - \min_x}{\max_x - \min_x}. \tag{A.1.3}$$

The two functions *scale255* and *rescale255* perform a more professional job. the function *scale255* shown in figure A.23 removes the bias of x. Also, if the maximum of x in this situation is more than 255, *scale255* declines the dynamic range. The

normalize

```
function J=normalize(I)
%Function normalize, normalize the data I.
%Usage:      J=normalize(I)
%Inputs:     I:Original image.
%Outputs:    J:Result.
%Revision 1.1 Arash Abadpour Summer 2004 (1383/6/25)
```

Figure A.21: Implementation of the *normalize* function.

normalize255

```
function J=normalize255(I)
%Function normalize255, normalize the data I.
%Usage:      J=normalize255(I)
%Inputs:     I:Original image.
%Outputs:    J:Result.
%Revision 1.0 Arash Abadpour Spring 2004 (1383/2/27)
```

Figure A.22: Implementation of the *normalize255* function.

function *scale255* returns the bias and gain. As such, using the function *rescale255* one can inverse the transformation (see Figure A.24).

The function *percentile* in **CIP** gives the value of x satisfying a certain level of CDF. This function is suitable for dynamic determination of thresholds (see Figure A.25). Similarly, the function *themost* returns the most frequently repeated member of a given array (the mode) (see Figure A.26). The two functions *energy* and *ExtractBitPlanes* compute the energy content of eigenimages and extract the

scale255

```
function [J,bias,scale]=Scale255(I)
%Function scale to 255 performs the scaling and shifting (only making
%small) to map I into the 0..255 interval.
%Usage:[J,bias,scale]=Sscale255(I);
%Inputs:     I:Input.
%Outputs:    J:Result.
%            bias, scale:Bias and Scale.
%Version 1.0, Arash Abadpour Winter 2005 (Feb. 23, 2005)
```

Figure A.23: Implementation of the *scale255* function.

rescale255

```
function J=Rescale255(I,bias,scale)
%Function rescale performs rescaling.
%Usage:J=Rescale255(I,bias,scale);
%Inputs:     I:Input.
%            bias, scale:Bias and Scale.
%Outputs:    J:Result.
%Version 1.0, Arash Abadpour Winter 2005 (Feb. 23, 2005)
```

Figure A.24: Implementation of the *rescale255* function.

percentile

```
function x=percentile(X,P,precision)
%Function percentile gives the values x in the array X for which P percent
%of data in X are smaller than x.
%Usage: x=percentile(X,P,precision);
%Inputs:     X:Data
%            P:Percentage.
%            precision:Precision
%Outputs:    x:Result.
%Revision 1.1 Arash Abadpour Spring 2004 (1383/2/6)
```

Figure A.25: Implementation of the *percentile* function.

bit–planes of a grayscale image, respectively (see Figures A.27 and A.28).

A.2 Visualization functions

CIP contains many function designed specifically for visualization purposes. For example, the function *ishow* which is shown in Figure A.29 is helpful for visualizing a fuzzy image or a mask. It also produces the result as an image, and not on a figure window, if the *show* argument is set to unity. This is helpful when one needs to save the results in a file.

For applications like fuzzy segmentation that a 3–D array contains the membership of pixels to different clusters, function *ifshow* is suited to show a membership plane using its index (see Figure A.30). The two functions *showlayer* and *showlayers* use also the information in the original image to visualize a single layer or all the

themost

```
function x=themost(X,precision)
%Function the most gives the value of X that has happened more.
%Usage: x=themost(X,precision);
%Inputs:     X:Data
%            precision:Precision
%Outputs:    x:Result.
%Revision 1.0 Arash Abadpour Winter 2004
```

Figure A.26: Implementation of the *themost* function.

Energy

```
function [e1,e2,e3]=energy(x1,x2,x3)
%Function energy compute sthe energy distribution.
%
%Usage:    [e1,e2,e3]=energy(x1,x2,x3);
%          [e1,e2,e3]=energy(X)
%Inputs:   x1,x2,x3 or X:Source[s].
%
%Outputs: e1,e2,e3:Energy ratio.
%Revision 1.1 Arash Abadpour Winter 2005 (Jan. 26, 2004)
```

Figure A.27: Implementation of the *Energy* function.

ExtractBitPlanes

```
function [Js,Es,P]=ExtractBitPlanes(I)
%Function extract bit planes extract the bit planes of I.
%Usage:[Js,Es,P]=ExtractBitPlanes(I);
%Inputs: I:The image.
%Outputs: Js: The planes.
%        Es:Successive approximations.
%        P:Successive values of PSNR.
%Revision 1.0, Arash Abadpour, Spring 2005 (May 16, 2005).
```

Figure A.28: Implementation of the *ExtractBitPlanes* function.

ishow

```
function J=ishow(I,M,bck,show)
%Function Ishow shows an image using theimshow function, except for the
%point that it performs the typecasting first.
%Usage:     J=ishow(I,M,bck,show)
%Inputs:    I:Input image.
%           M:Mask image.
%           bck:Background
%           show: Show?
%Outputs:   J:result;
%Revision 1.5 Arash Abadpour Spring 2005 (April 16, 2005)
```

Figure A.29: Implementation of the *ishow* function.

ifshow

```
function ifshow(F,k)
%Function image fuzzy show extracts one of the images in the 3-D image F
%and shows it on the screen.
%Usage: ifshow(F,k);
%Inputs:     F:Fuzzy images.
%            k:index.
%Outputs:
%Revision 1.0 Arash Abadpour Spring 2004
```

Figure A.30: Implementation of the *ifshow* function.

layer, respectively (see Figure A.31 and Figure A.32). When the borders of a crisp segmentation are known *showsegmentation* (see Figure A.33) would be the proper solution.

Function *showqrs* shows a set of query regions in a color image (see Figure A.34). Functions *showr*, *showrb*, *showrc*, *showrgui*, *showrordr*, *showcomposition*, and *segresult* other functions designed for visualizing the results of segmentation (see Figures A.35, A.36, A.37, A.38, A.39, A.40, and A.41).

Functions *hist3d* and *showcluster* are suited to visualize the results of segmentation in the color cube (see Figures A.42 and A.43). Finally, the two functions *showcolor* and *showcolors* are designed to visualize an individual color or a set of colors (see Figures A.44 and A.45). These functions are utilized to produce the visualizations shown in this thesis.

ShowLayer

```
function [Ic,If]=ShowLayers(Fs,i,I)
%Function show layer, shows the i-th layer.
%Usage:  [Ic,If]=ShowLayers(Fs,i,I)
%Inputs:        Fs:Fuzzy regions
%              i:Index.
%              I:Original image.
%Outputs:       Ic,If:Result.
%Revision 1.0 ArashAbadpour Spring 2004
```

Figure A.31: Implementation of the *ShowLayer* function.

ShowLayers

```
function Ic=ShowLayers(Fs,X,fuzzy)
%Function show layers, shows the layers in X.
%Usage:  Ic=ShowLayers(Fs,X,fuzzy)
%Inputs:        Fs:Fuzzy regions
%              X:Composition
%              fuzzy:2 or 1 or 0
%Outputs:       Ic:Result
%Revision 1.21 ArashAbadpour Winter 2004
```

Figure A.32: Implementation of the *showlayers* function.

ShowSegmentation

```
function ShowSegmentation(X,Xs,Ys,I)
%Function show segmentation, shows the segmentation result.
%Usage:   ShowSegmentation(X,Xs,Ys)
%Inputs:         X:Composition
%               Xs,Ys:Snakes
%               I:Original image
%
%Outputs:
%Revision 1.0 ArashAbadpour Winter 2004 (1383/1/30)
```

Figure A.33: Implementation of the *ShowSegmentation* function.

Showqrs

```
function Showqrs(I,R)
%Show shows query regions shows the image I and the qurey region in r.
%
%Usage:Showqrs(I,R)
%Inputs:     I:The image.
%            R:Query regions (R(k,:)=(y1,x1,y2,x2)).
%Outputs:
%Revision 1.0 Arash Abadpour Fall 2004 (Oct. 26, 2004)
```

Figure A.34: Implementation of the *Showqrs* function.

showr

```
function IE=ShowR(r,Id,E)
%Show Regions visualizes the regions in r
%
%Usage:IE=showr(r,Id,E)
%Inputs:     r: Region information
%           Id: Double version of original image
%           E:Embeded information.
%Outputs:    IE: Visualized regions.
%Revision 2.01 (c) Arash Abadpour Spring 2004 (1383/2/22)
```

Figure A.35: Implementation of the *showr* function.

ShowRb

```
function IE=ShowRB(r,Id,er)
%Show Regions borders visualizes the regions in r
%
%Usage:IE=ShowRB(r,Id,er)
%Inputs:     r: Region information
%           Id: Double version of original image
%           er:Error margin.
%Outputs:    IE: Visualized regions.
%Revision 1.22 Arash Abadpour Summer 2004 (1383/5/14)
```

Figure A.36: Implementation of the *showrb* function.

showrc

```
function IE=ShowRc(r,Id,index)
%Show regions Component visualizes the selected component.
%
%Usage:IE=ShowRc(r,Id,index)
%Inputs:     r: Region information
%           Id: Double version of original image
%           index:desired componsnt.
%Outputs:    IE: Visualized component.
%Revision 1.0 Arash Abadpour spring 2004 (1383/2/27)
```

Figure A.37: Implementation of the *showrc* function.

ShowRGUI

```
function ShowRGUI(r,I,col)
%Show regions borders in a graphical user interface visualizes the regions in r
%
%Usage:ShowRGUI(r,Id)
%Inputs:     r: Region information.
%           I: The original image.
%           col:Color.
%Revision 1.0 Arash Abadpour Summer 2004 (1383/6/25)
```

Figure A.38: Implementation of the *ShowRGUI* function.

ShowRORDR

```
function IE=ShowRORDR(r,Id)
%Show Regions Order visualizes the regions in r using their orders
%
%Usage:IE=ShowRordr(r,Id)
%Inputs:     r: Region information
%           Id: Double version of original image
%Outputs:    IE: Visualized regions.
%Revision 1.0 (c) Arash Abadpour 1382
```

Figure A.39: Implementation of the *ShowRORDR* function.

ShowComposition

```
function [Ic,If]=ShowComposition(Fs,X,I)
%Function show composition shows the composition of Fs as described in X
%from the image I. The result is saved into If as a fuzzy membership and Ic
%as a colored image.
%Usage: [Ic,If]=ShowComposition(Fs,X,I);
%Inputs:        Fs:Fuzzy memberships
%               X:Composition
%               I:Original image
%Outputs:       Ic:Color reconstruction
%               If:Fuzzy reconstruction
%Revision 1.0 Arash Abadpour Winter 2004
```

Figure A.40: Implementation of the *ShowComposition* function.

segresult

```
function K=segresult(I,J,realdata)
%Function segmentation result renders the segmentation results in a single
%image.
%Usage:      K=segresult(I,J,realdata);
%Inputs:     I:Original image.
%            J:embership maps.
%            realdata:Wether to use real data in visualization.
%Outputs:    K:The rendered image.
%Revision 1.2 Arash Abadpour Spring 2005 (April 16, 2005)
```

Figure A.41: Implementation of the *segresult* function.

hist3d

```
function h=hist3d(I,col,step,M)
%function three dimensional histogram shows the 3D histogram of the image I
%Usage:      h=hist3d(I,col,step,M);
%Inputs:     I:Original image.
%            col:Color;
%            step:Image sampling ratio.
%            M:Mask.
%Outputs:
%Revision 1.211 Arash Abadpour Winter 2005 (Feb. 9, 2004)
```

Figure A.42: Implementation of the *hist3d* function.

showcluster

```
function showcluster(I,J,newfigure)
%Function show cluster shows the clusters of I using ML.
%Usage:showcluster(I,J)
%Inputs:     I:Original image.
%           J:Membershipmaps.
%           newfigure:Make a new figure?
%Outputs:
%Revision 1.01 Arash Abadpour Fall 2004 (Oct. 24, 2004)
```

Figure A.43: Implementation of the *showcluster* function.

ShowColor

```
function I=ShowColor(x,H,W)
%Function show color composes an image which shows the color in x
%Usage: I=ShowColor(x,H,W);
%Inputs:x:The color (3*1 or 1*3).
%       H,W:Image size (default: 128).
%Outputs:I:Result.
%Revision 1.0, Arash Abadpour, Spring 2005 (May 1, 2005).
```

Figure A.44: Implementation of the *ShowColor* function.

ShowColors

```
function I=ShowColors(x,H,W)
%Function show colors composes an image which shows the colors in x
%Usage: I=ShowColors(x,H,W);
%Inputs:x:The color (3*n or n*3).
%       H,W:Image size (default: 128).
%Outputs:I:Result.
%Revision 1.0, Arash Abadpour, Spring 2005 (May 1, 2005).
```

Figure A.45: Implementation of the *ShowColors* function.

A.3 Data files

There are some *.mat* files included in CIP. Here is a list of them:

- *Swatch.mat*: It contains the information regarding the homogenous swatch bank used in FPCA–based direct copy protection discussed in Section 3.8.1.

- *BLUTBest.mat*, *BLUTCIE-a.mat*, *BLUTCIE-u.mat*, *BLUTCr.mat*, *BLUTCro.mat*, *BLUTHSI.H.mat*, *BLUTMin.mat*, *BLUTQo.mat*, *BLUTRg.mat*, *BLUTRg+.mat*, *BLUTVo.mat*, *BLUTYIQ.I.mat*, and *BLUTYUV.V.mat*: BLUTs of skin (see Section 3.11.1).

- *LUTBest.mat*, *LUTCbCr.mat*, *LUTCIE-Luv.mat*, *LUTHMMD.mat*, *LUTHSI.mat*, *LUTHSV.mat*, *LUTI1I2I3.mat*, *LUTIQ.mat*, *LUTNbr.mat*, *LUTRGB.mat*, *LUTTSL.mat*, and *LUTYUV.UV.mat*: LUTs of skin (see Section 3.11.1).

LaplacianWeight

```
function W=LaplacianWeight(I,ds,m,alpha,beta)
%Function laplacian weights compute sthe laplacian weights of the image I.
%Usage:W=LaplacianWeight(I,ds,m,alpha,beta);
%Inputs:I:Original Image.
%       ds:Downsampling ratio.
%       m:Fuzziness.
%       alpha:Alpha of Laplacian (default:0.2).
%       beta:Afterwards averaging disk radius (default:0).
%Outputs:W:Weight map.
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 9, 2005)
```

Figure A.46: Implementation of the *LaplacianWeight* function.

- *SSa.mat*, *SSH.mat*, and *SSI_3.mat*: Best χ^2–based signatures of skin (see Section 3.11.2).

- *PCASkin.mat*: Information regarding the principal skin (see Section 3.11.3).

A.4 Methods Not Described in Chapter 3

The set of functions *CR_CONSTANTS*, *CRPCA_CONSTANTS*, *LaplacianWeight*, *RCEput*, *RCEWput*, *RCFPCAPallet*, *RCPallet*, *RCPCAev*, *RCPCAPallet*, *RCPCAput*, *RCput*, *RCSegment*, *RCWPallet*, *ShowPallet*, *ShowPallet3D*, *ShowPalletTree* perform color quantization and not discussed in this thesis.

RCEput

```
function [J,index]=RCEput(I,Y)
%Function reduce color Euclidean put performs exactly what the name
%implies.
%Usage:[J,index]=RCEput(I,Y);
%Inputs:I:Original image.
%       Y:The pallet (3*n).
%Outputs:J:Result.
%            index:Index map.
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 8, 2005)
```

Figure A.47: Implementation of the *RCEput* function.

RCEWput

```
function [J,index]=RCEWput(I,Y,m1,m2,radius)
%Function reduce color Euclidean weighted put performs exactly what the name
%implies.
%Usage:[J,index]=RCEWput(I,Y,m1,m2,r);
%Inputs:I:Original image.
%       Y:The pallet (3*n).
%       m1,m2,r:Fuzziness.
%       radius:Averaging disk radius.
%Outputs:J:Result.
%            index:Index map.
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 9, 2005)
```

Figure A.48: Implementation of the *RCEWput* function.

RCFPCAPallet

```
function [T,P,t]=RCFPCAPallet(I,ds,N,m,FullTree)
%Function reduce color using FPCA performs what the na,e implies.
%Usage:[T,P]=RCFPCAPallet(I,ds,N,m,FullTree);
%Inputs:I:Original Image.
%       ds:Downsampling ratio.
%       N:Colors number.
%       m:Emphasize.
%       FullTree:Finding the full tree? (default : 0)
%Outputs:T:The tree.
%        P:The pallet.
%        t:Elapsed time.
% Revision 2.01 Arash Abadpour Spring 2005 (May 5, 2005)
```

Figure A.49: Implementation of the *RCFPCAPallet* function.

RCPallet

```
function [T,P,t]=RCPallet(I,ds,C,m,delta)
%Funciton reduced color pallet computes the color pallet of I with the
%given parameters.
%Usage:[T,P,t]=RCPallet(I,ds,C,m,delta);
%Inputs:I:Original Image.
%       ds:Downsampling ratio.
%       C:Pallet size.
%       m:Fuzziness (leave it unity!).
%       delta:Convergence Condition (5 results satisfactorily)
%
%Outputs:T:The tree.
%        P:The pallet.
%        t:Elapsed time.
%Revision 1.03 Arash Abadpour Winter 2005 (Feb. 8, 2005)
```

Figure A.50: Implementation of the *RCPallet* function.

RCPCAev

```
function [e,v]=RCPCAev(T,k)
%Function reduce color using PCA, eta and v, gives the eta and v matrices
%of the i–th node. It is internally used in other RCPCA functions. it is
%not devisable to use it if you are not completely sure that you know what
%it does.
%Usage:[e,v]=RCPCA(T,k);
%Inputs: T:The tree.
%       k:Index.
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 8, 2005)
```

Figure A.51: Implementation of the *RCPCAev* function.

RCWPallet

```
function [T,P,t]=RCWPallet(I,ds,C,m,delta)
%Funciton reduced color weighted pallet computes the color pallet of I with the
%given parameters.
%Usage:[T,P,t]=RCWPallet(I,ds,C,m,delta)
%Inputs:I:Original Image.
%       ds:Downsampling ratio.
%       C:Pallet size.
%       m:Fuzziness (leave it unity!).
%       delta:Convergence Condition (5 results satisfactorily)
%
%Outputs:T:The tree.
%        P:The pallet.
%        t:Elapsed time.
%Revision 1.02 Arash Abadpour Winter 2005 (Feb. 8, 2005)
```

Figure A.52: Implementation of the *RCWPallet* function.

RCPCAPallet

```
function [T,P,t]=RCPCAPallet(I,ds,N,FullTree)
%Function reduce color using PCA performs what the na,e implies.
%Usage:[T,P,t]=RCPCAPallet(I,ds,N,FullTree);
%Inputs:I:Original Image.
%       ds:Downsampling ratio.
%       N:Colors number.
%       FullTree:Finding the full tree? (default : 0)
%Outputs:T:The tree.
%        P:The pallet.
%        t:Elapsed time.
%Revision 2.01 Arash Abadpour Winter 2005 (March 3, 2005)
```

Figure A.53: Implementation of the *RCPCAPallet* function.

RCPCAput

```
function [J,index]=RCPCAput(I,T,P)
%Function reduce color PCA-based put performs exactly what the name
%implies.
%Usage:[J,index]=RCPCAput(I,T,P);
%Inputs:I:Original image.
%           T:The tree.
%           P:The pallet.
%Outputs:J:Result.
%             index:Index map.
%Revision 2.0 Arash Abadpour Winter 2005 (Feb. 8, 2005)
```

Figure A.54: Implementation of the *RCPCAput* function.

ShowPallet

```
function I=ShowPallet(P,w,W,h)
%Function show pallet shows thge pallet P.
%Usage: I=ShowPallet(P,w,W,h);
%Inputs: P:Pallet (3*n).
%        w,h:Sample sizes.
%        W:Image size.
%Output:I:Result.
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 6, 2005)
```

Figure A.55: Implementation of the *ShowPallet* function.

RCput

```
function [J,index,t]=RCput(I,T,P,method,weighted)
%Funcitonr educed color put puts the color contents of T and P into I to
%produce J.
%Usage:[J,index,t]=RCput(I,T,P,method,weighted)
%Inputs:I:Original image.
%       T,P:Results of reducing the color.
%       method: 1: Do not generalize the pallet to those having proper
%                  representatives.
%               2: Start from generalizaed pallet.
%       weighted: Use weighted Manhatan distance.
%Output:J:Result.
%       index:Index map.
%       t:Elapsed time.
%Revision 1.01 Arash Abadpour Winter 2005 (Feb. 7, 2005)
```

Figure A.56: Implementation of the *RCput* function.

ShowPallet3D

```
function ShowPallet3D(P)
%Function show pallet shows thge pallet P.
%Usage: ShowPallet(P);
%Inputs: P:Pallet (3*n).
%Output:
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 9, 2005)
```

Figure A.57: Implementation of the *ShowPallet3D* function.

RCSegment

```
function [J,J1,index,index1,Y,Y1]=RCSegment(I,refine,radius,m1,m2,N,ds,delta)
%function reduce color segmente performs color image segmentation using
%color reduction thecniques.
%Usage:[J,J1,index,index1,Y,Y1]=RCSegment(I,refine,radius,m1,m2,N,ds,delta);
%Inputs:I:Original Image
%Outputs:
%Revision 1.0 Arash Abadpour Winter 2005 (Feb. 9, 2005)
```

Figure A.58: Implementation of the *RCSegment* function.

ShowPalletTree

```
function fig=ShowPalletTree(T,ShowLabel)
%Function show pallet shows the tree T.
%Usage: fig=ShowPalletTree(T,ShowLabel);
%Inputs: T:The tree.
%        ShowLabel: default is 0.
%Output:fig:The handle to the resulting figure.
%Revision 1.1 Arash Abadpour Winter 2005 (March 2, 2005)
```

Figure A.59: Implementation of the *ShowPalletTree* function.

Appendix B

Sample Images

CIP package includes a set of samples files, most of which are standardized materials. In this section, an icon of all sample images is visualized. Also, the original owners are acknowledged. Many of the sample images are captured by the author. These images are photographed by an *A60 Canon* (2 Mega Pixels) camera in daylight or with flash and then cropped and resized. Also, there are a few images with unidentified owners. These images are those found in Internet resources with no indication of the copyright holders. We have done our best to identify the original owner of each sample image included in **CIP** and to include no copyrighted material, unless the owners have promised us to do so.

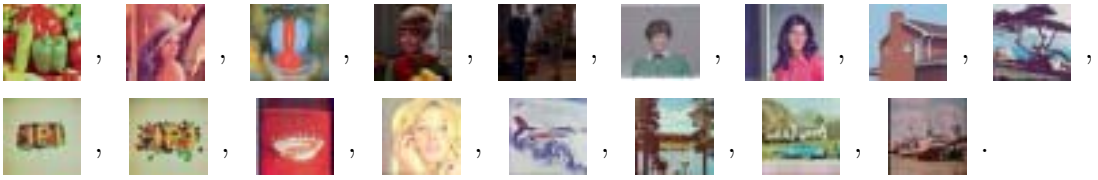
For the images captured by the author's camera, more than ninety percent of the situations the Automatic settings are used. Also, the camera is set to capture at 1600×1200 with the *superfine* quality. As such, a single image occupies about 800KB and is originally saved in jpeg format by the camera firmware. For the images

acquired from the Internet, firstly they are converted to the jpeg format (with the largest available quality of 12 in *Adobe Photoshop*). Then all the images are converted to 512×512 pixels. This conversion is always down–sampling, except for a very few standard images which are originally captured in a lower resolution.

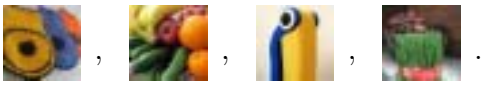
B.1 Color Images

The following color images are included in **CIP**.

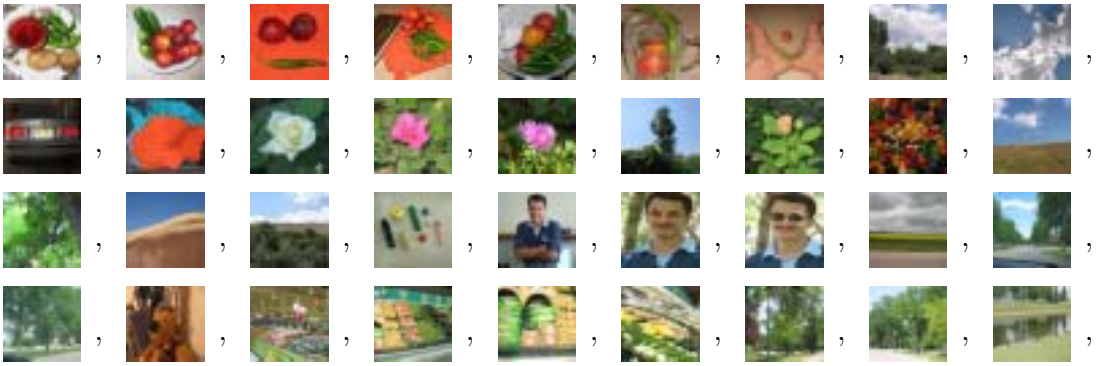
- Signal and Image Processing Institute at the University of Southern California (USC-SIPI):

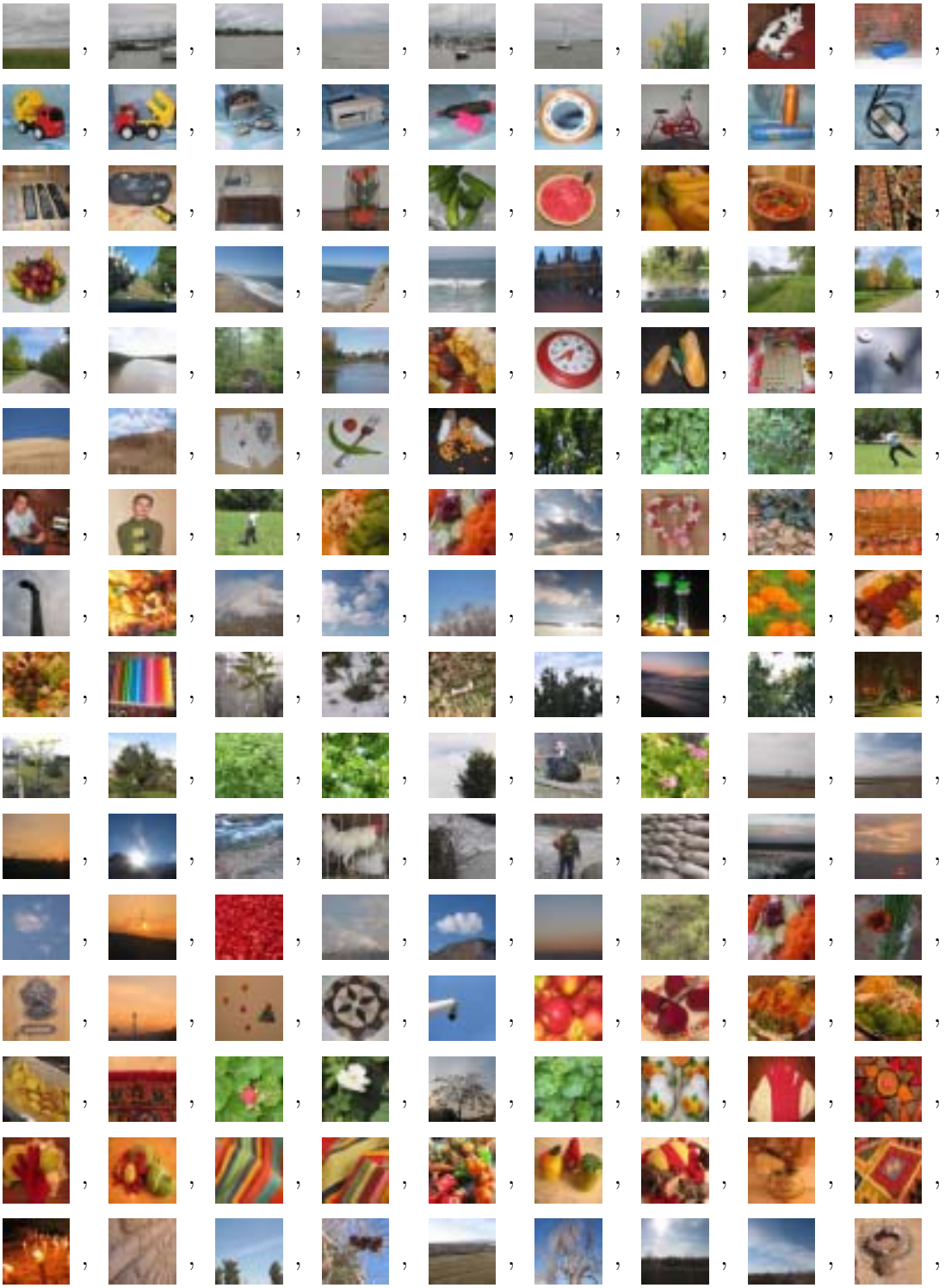


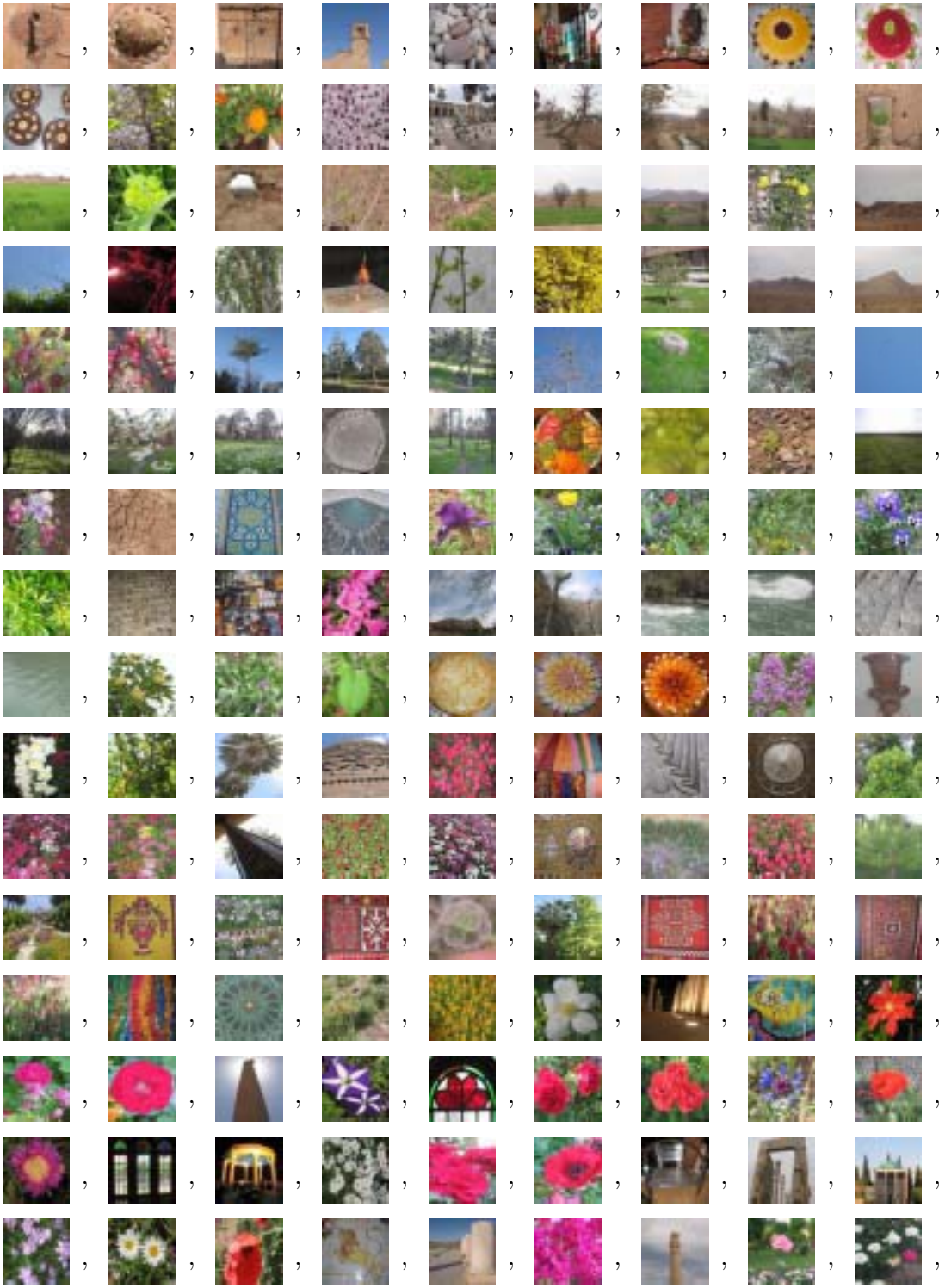
- *S. Tabatabaii S.* and *A. Qanavati*:

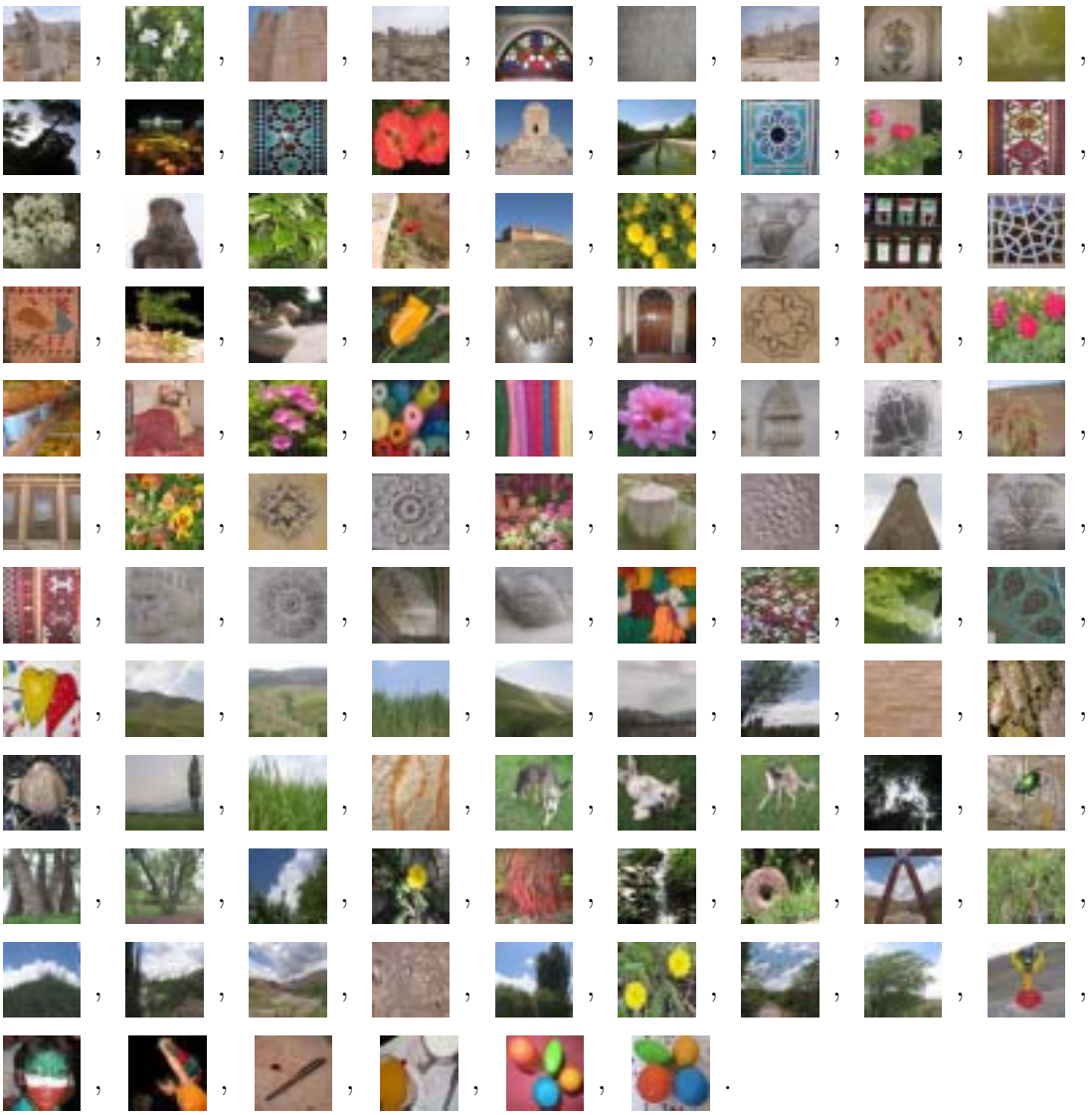


- Photographed by the author:



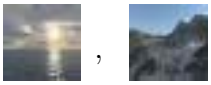






- National Oceanic and Atmospheric Administration Photo Library, adopted from [297]:

- *Simon Premoze*, adopted from [297]:



- *Erik Reinhard*, *Michael Ashikhmin*, *Bruce Gooch*, and *Peter Shirley* [297]:

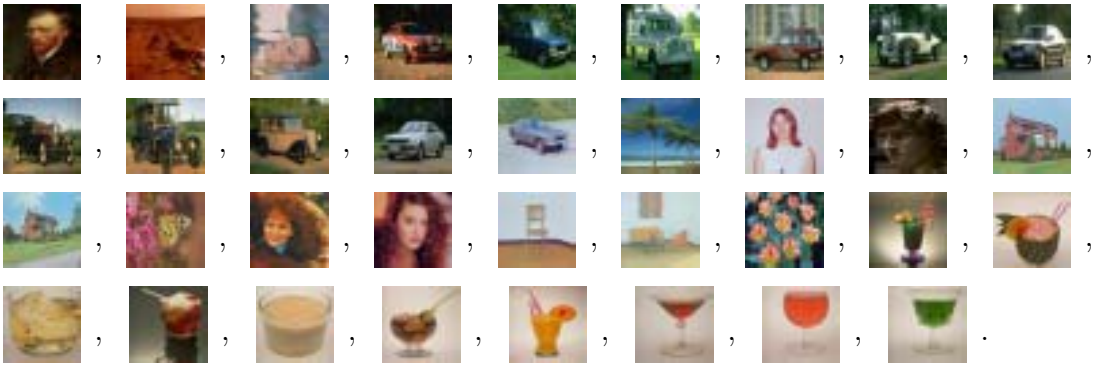
.

- Arles, September 1888, oil on canvas; image from the *Vincent van Gogh* Gallery, *www.vangoghgallery.com*, adopted from [297]:
  .
- *Oliver Deussen*, adopted from [297]:
  , .
- NOAA Photo Library, adopted from [297]:
  .
- *www.artchive.com*, adopted from [156]:
  , .
- *Mike Briner*, *mbphoto@spraynet.com*:
  .
- *Brent Reed*, *brent@reedservices.com*:
  .
- *Youngha Chang*, *Surguru Saito*, and *Masayuki Nakajima* [92]:
  , .

- *ultima.cs.unr.edu/getimages.htm*:

.

- The Ohio State University:

 .

- *Toni Lankerd*:

, .

- *Jaime Vives Piqueres*, *www.ignorancia.org/index.shtml*, *jaime@ctav.es*:

.

- *Kevin Odhner*, *jko@home.com*:

.

- *Robert E. Barber*, Barber Nature Photography, *rebarber@msn.com*:

, , , , , .

- *Grald Deshais*:

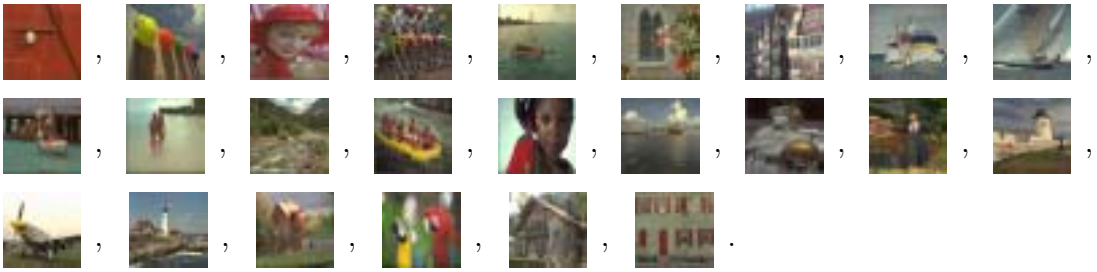
.

- *Karel de Gendre*, *www.profotos.com/pros/profiles/index.cfm?member=192*:
  , , .
- *Eric Laboure*:
  .
- *F. A. P. Petitcolas*, *www.petitcolas.net/fabien/*:
  , .
- *Patrick Loo*, University of Cambridge, *patrickl@autonomy.com*:
  , .
- *Ahmed Desoky*, *www.spd.louisville.edu/ ahdeso01*:
  .
- Unidentified Owner:
  , , , , , , , , ,
  , , , , , , , , ,
  , , , , , , , .
- IBM Madrid S.C.:
  .
- National Institute of Mental Health, Research Services Branch, Image Database:
  , , , , , .

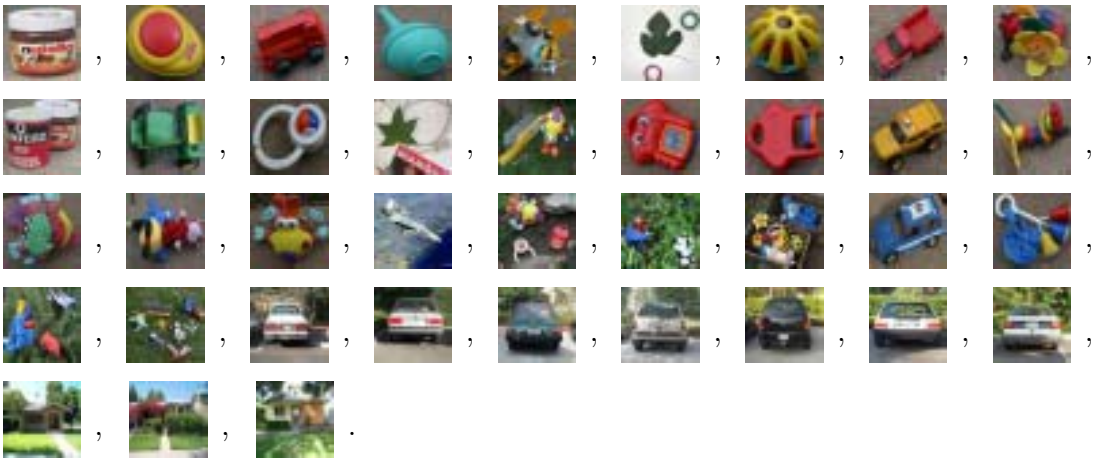
- *r0k.us/graphics/images/GreatWave.png*:

.

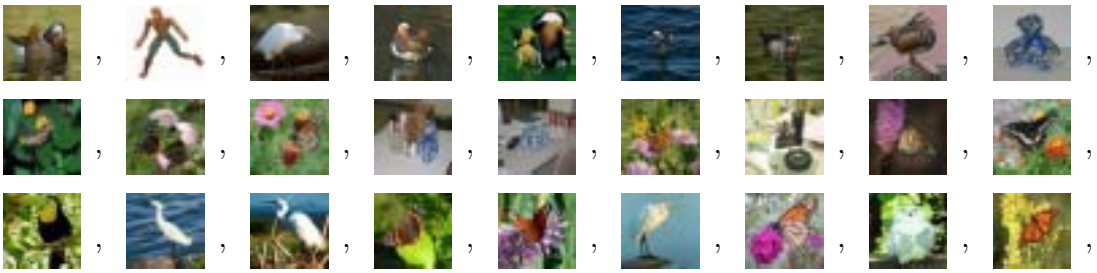
- Kodak PhotoCD (PCD0992):


.

- Computational Vision at Caltech image database, *www.vision.caltech.edu/*:


.

- Ponce Research Group Datasets for Computer Vision Research, *www-cvr.ai.uiuc.edu/ ponce_grp/data*:



, .

- mPower3/Emerge:

  , .

- *Susan Cohen*:

  , .

- Anglo–Australian Observatory, *David Malin*:

  .

- *J. C. Russ* [307]:

  , .

- *www.mathworks.com*:

  , , , , , , , , .

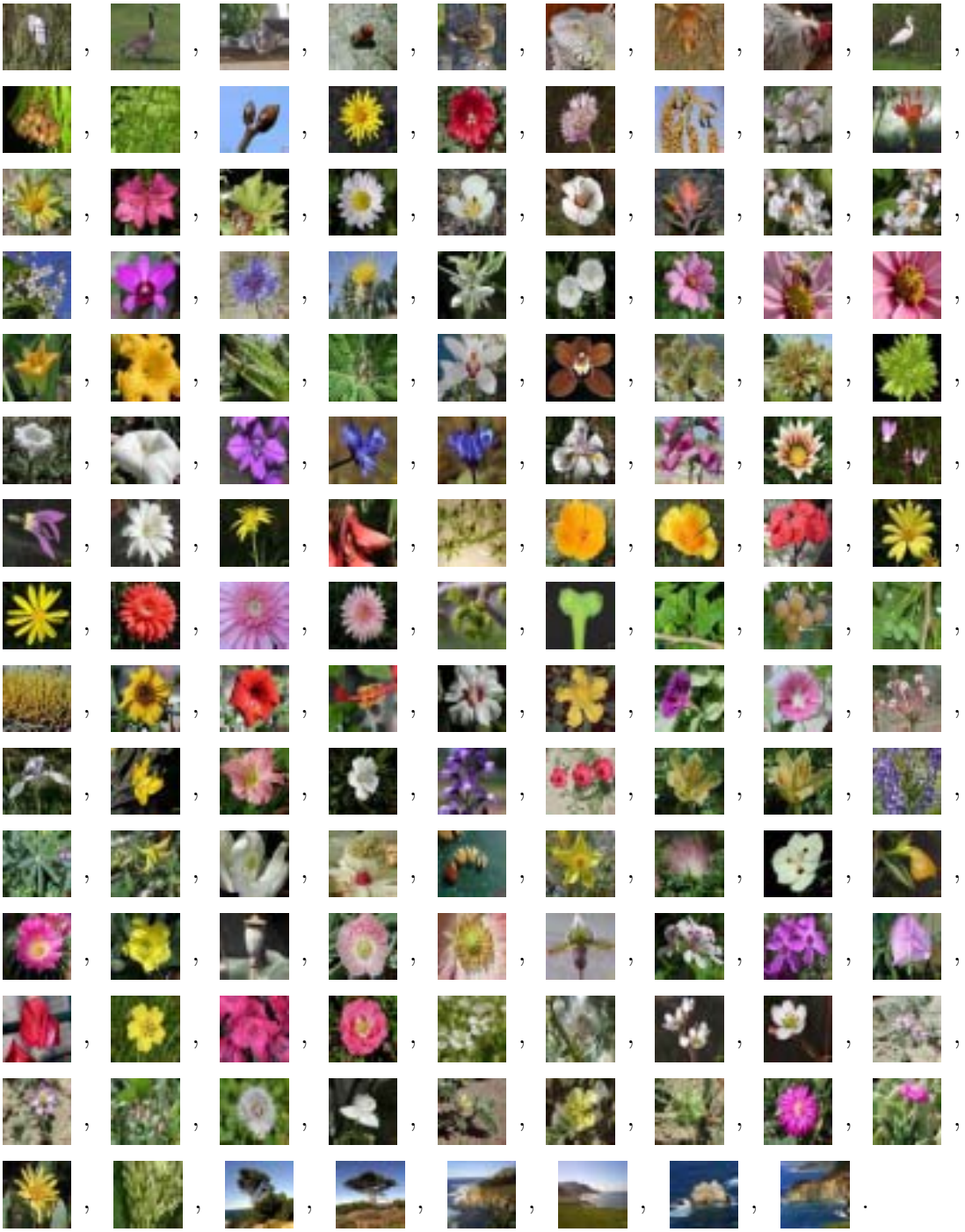
- Wiry Object Recognition Database, Carnegie Mellon University, *www.cs. cmu.edu/ ∼owenc/ word.htm*:

  .

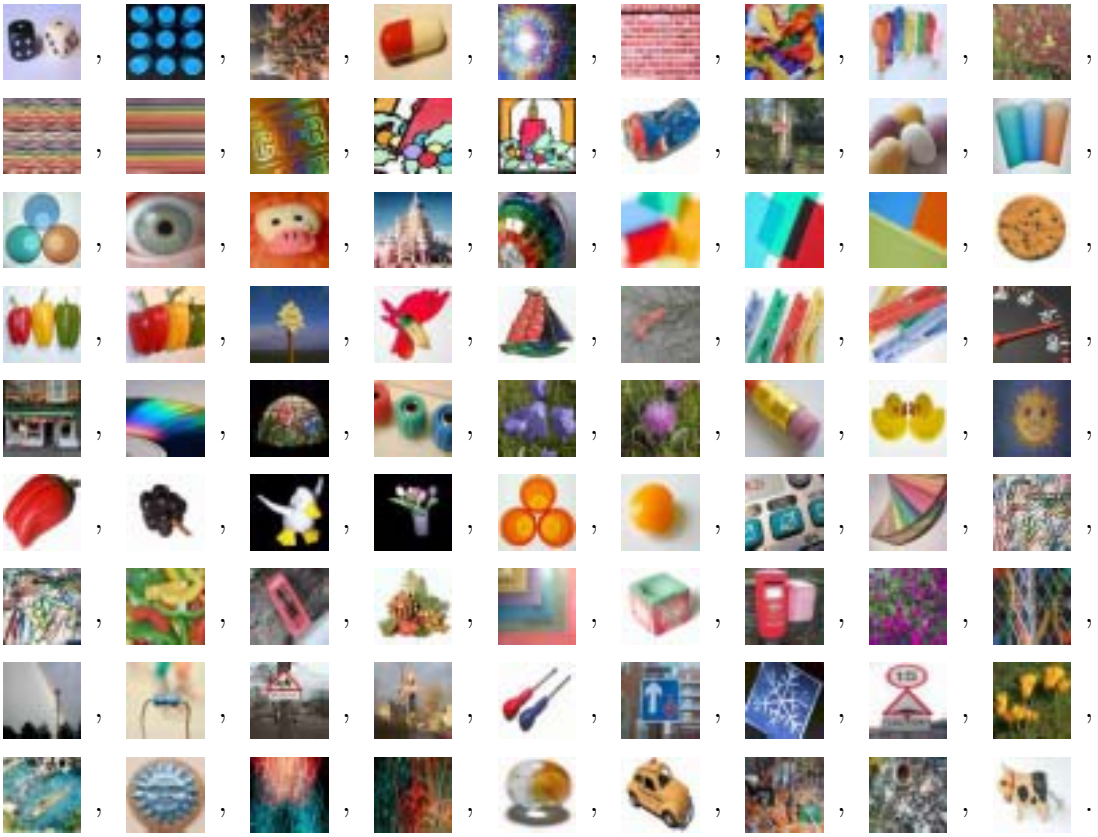
- *www.imageafter.com*:

  , , , .

- Steve's Digital Photos, *arnica.csustan.edu/photos/index.htm*:

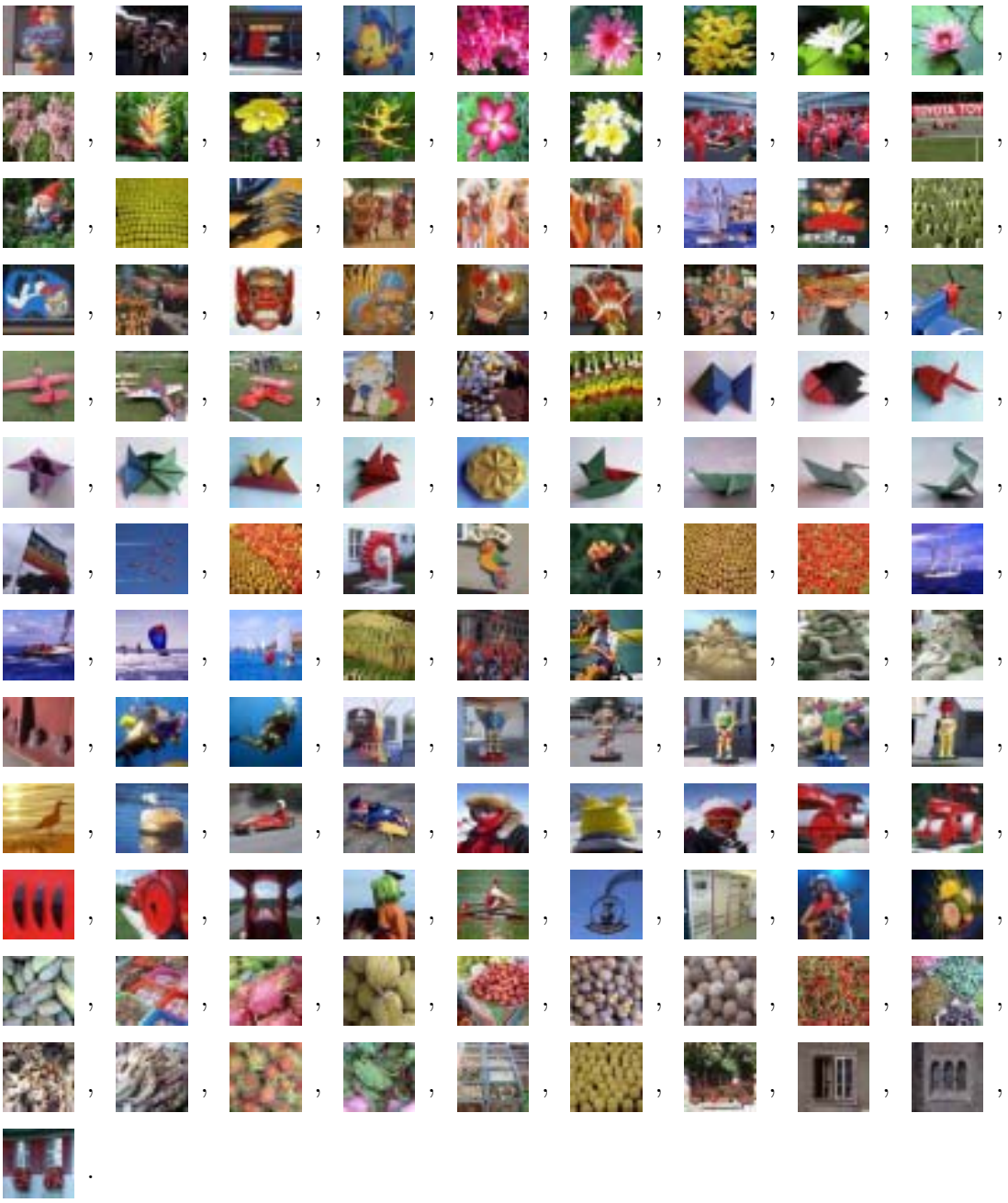


- *www.freeimages.co.uk*:

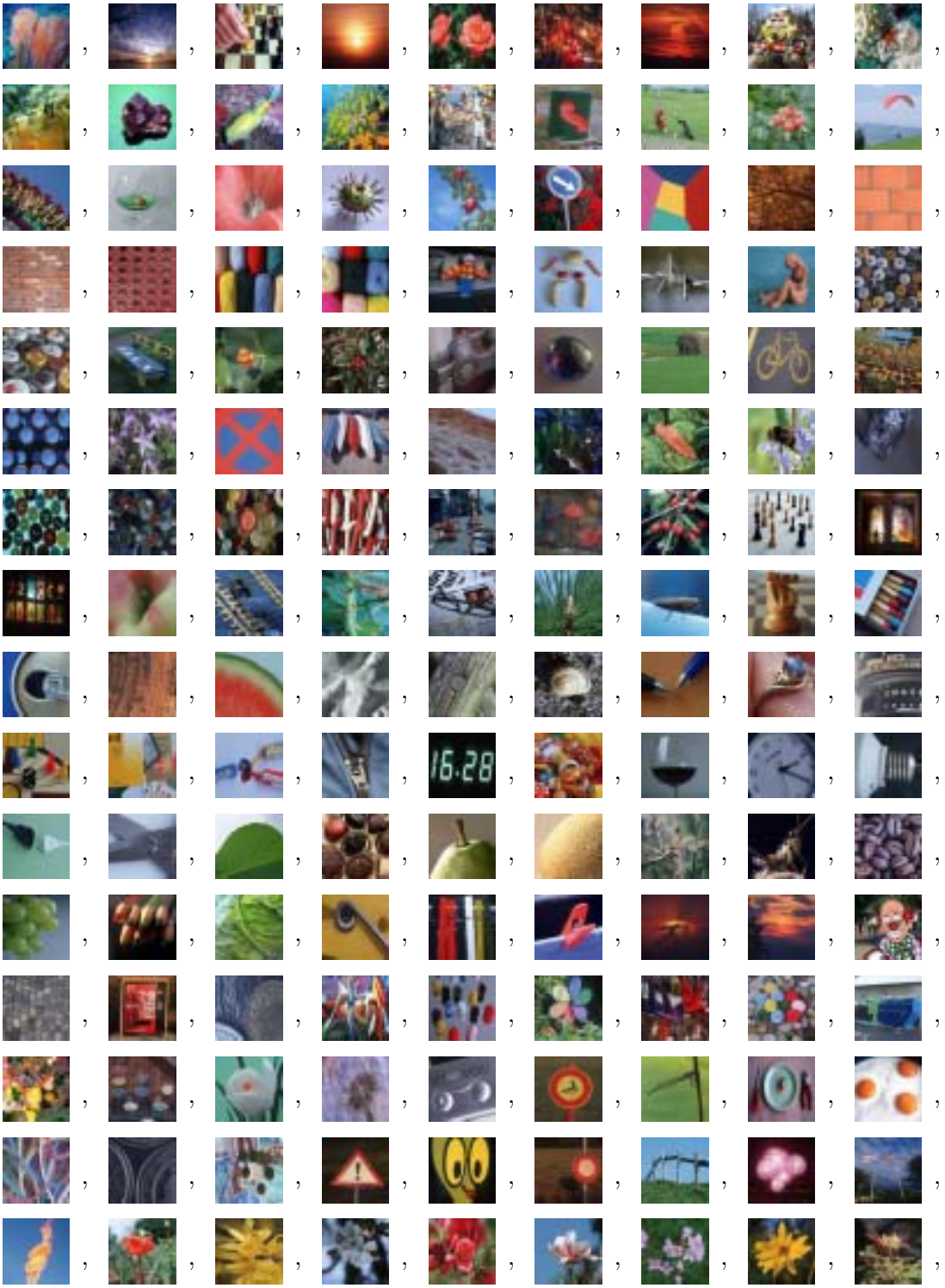


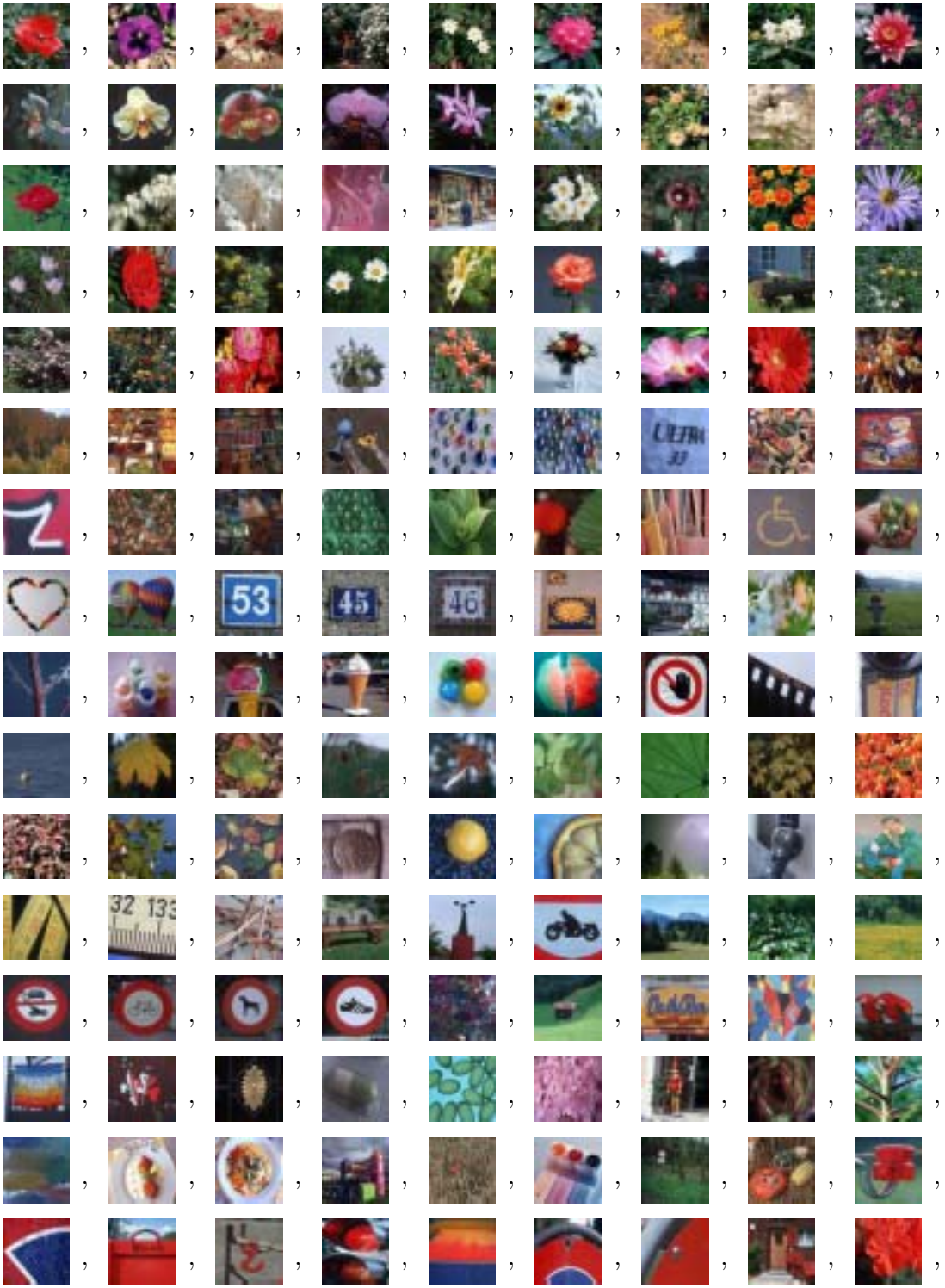
- *www.free-pictures-photos.com*:

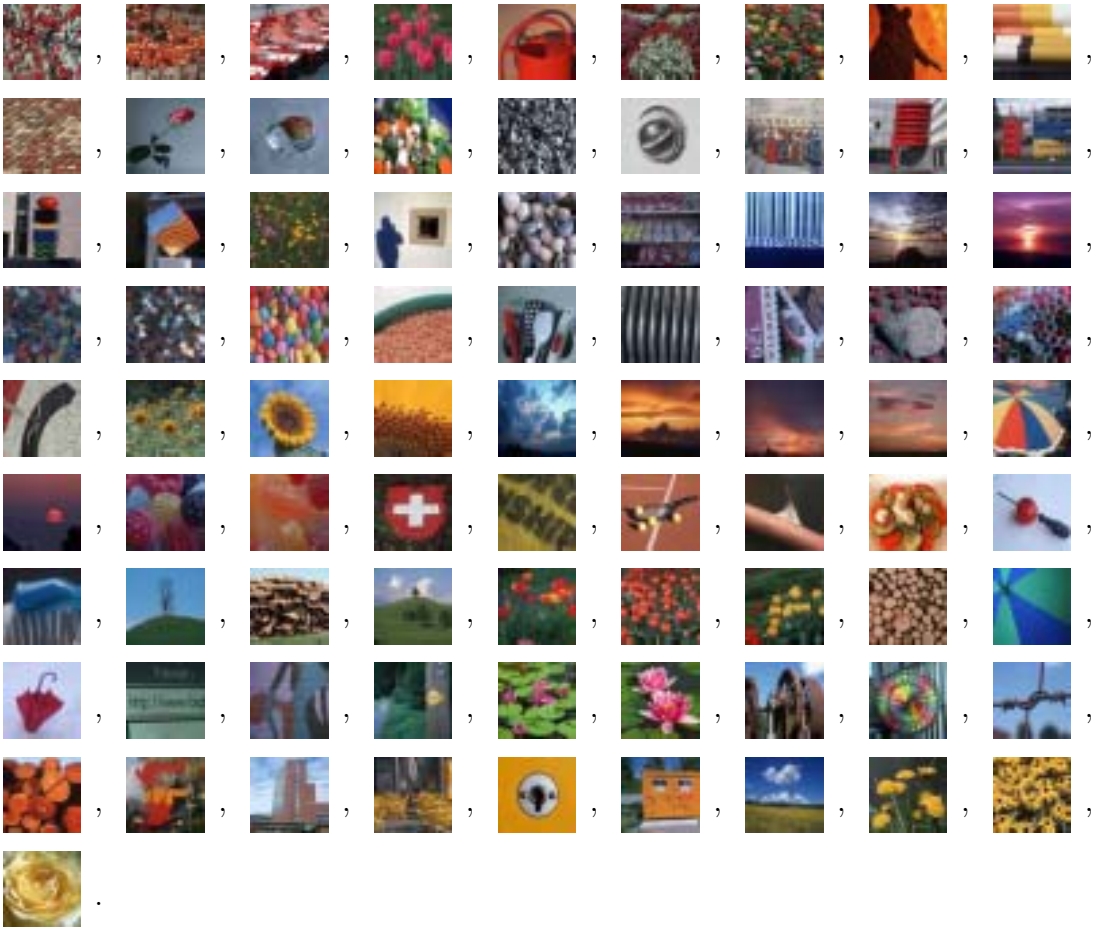
- *www.picture-newsletter.com*:



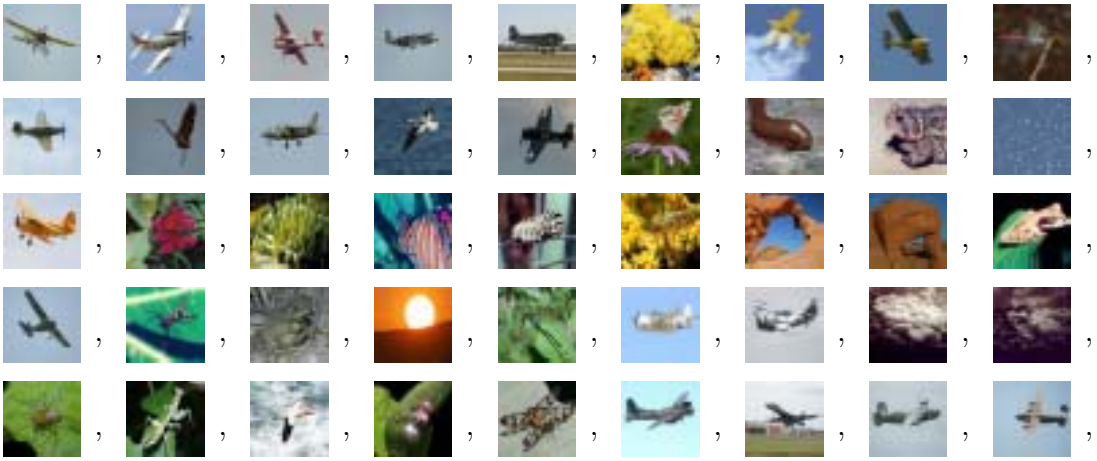
- *www.bigfoto.com*:

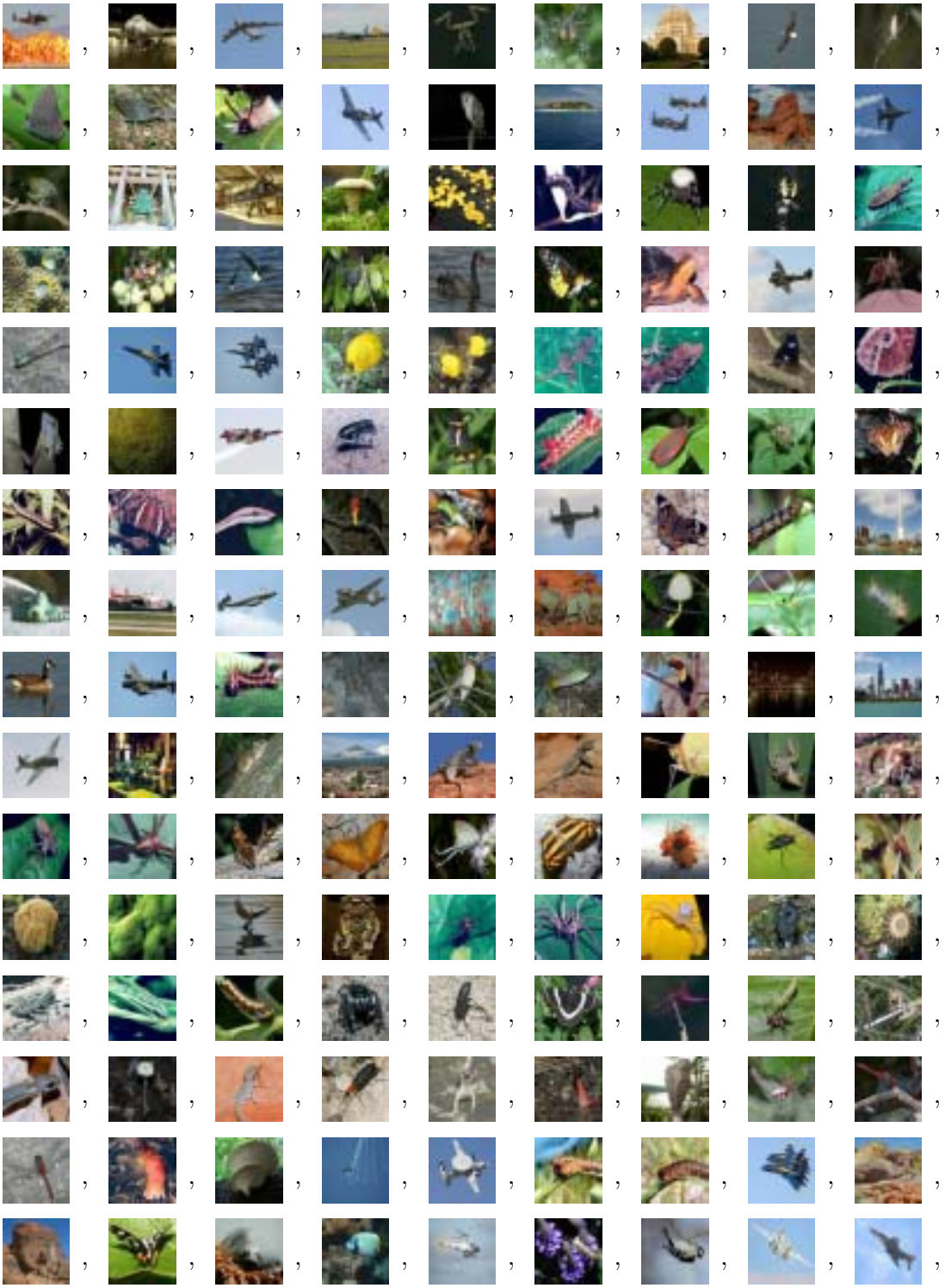

16:28

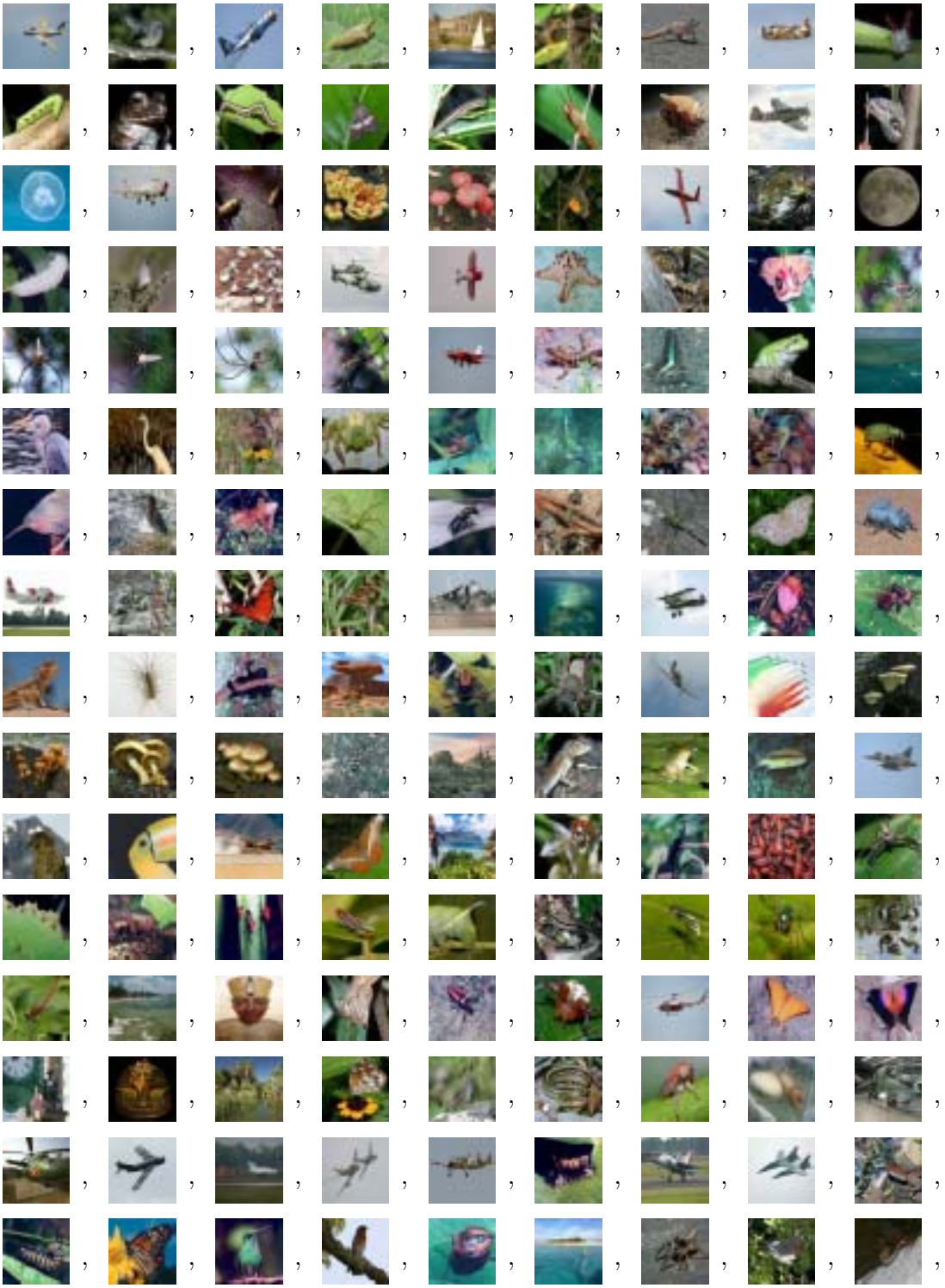


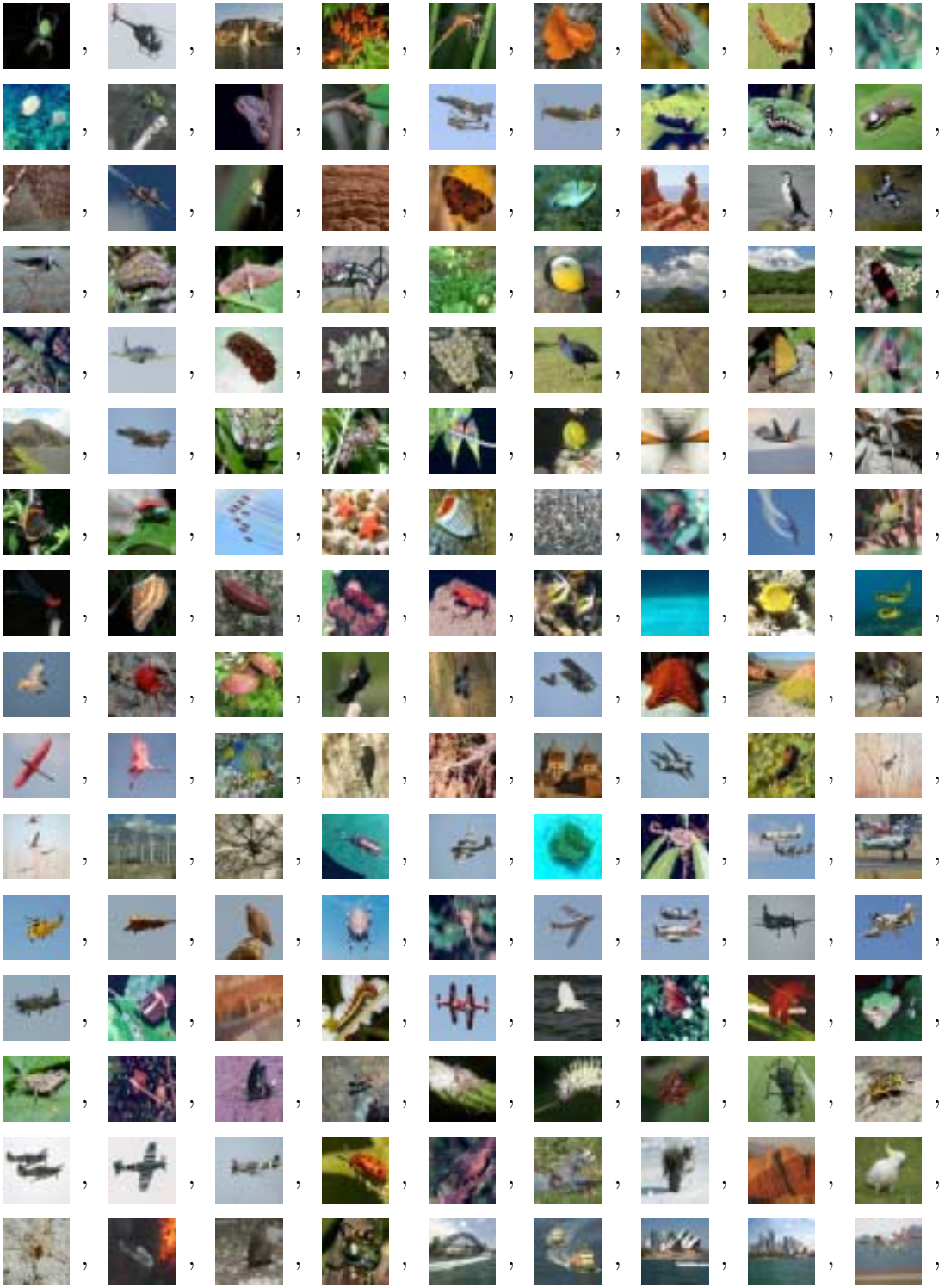


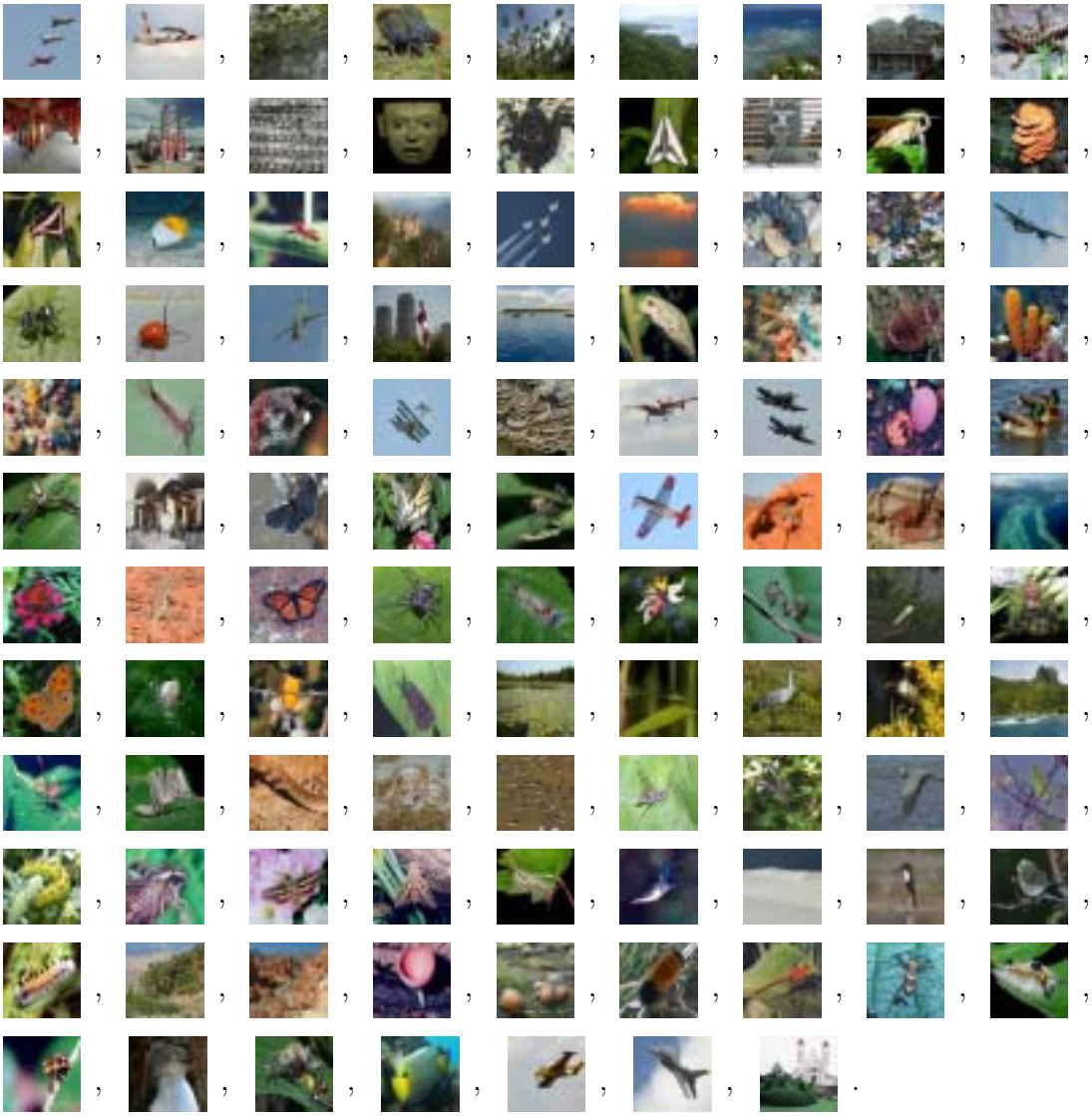
- *Richard Seaman, www.richard-seaman.com/*:











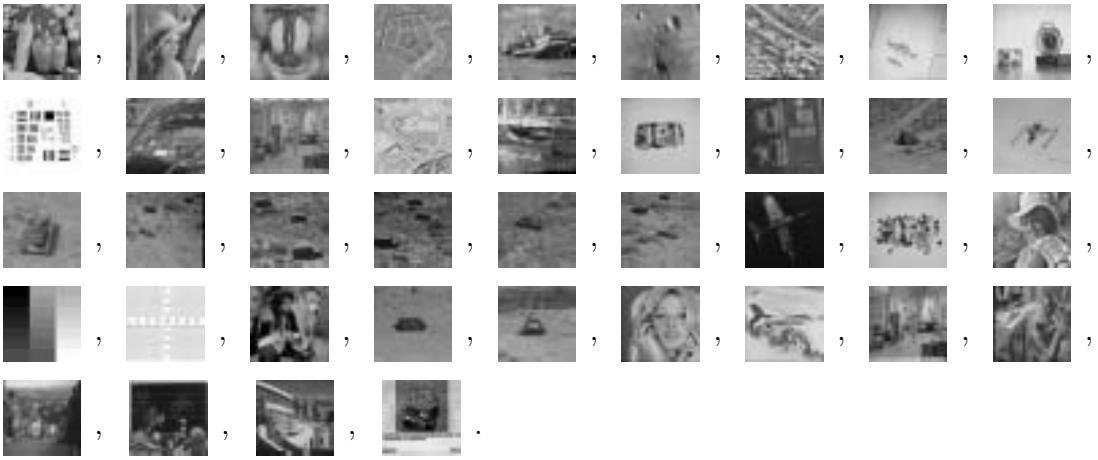
B.2 Grayscale Images

The following grayscale images are included in **CIP**.

- Institutes:

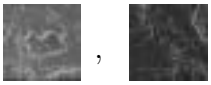
.

- Signal and Image Processing Institute at the University of Southern California (USC-SIPI):



- *ultima.cs.unr.edu/getimages.htm*:

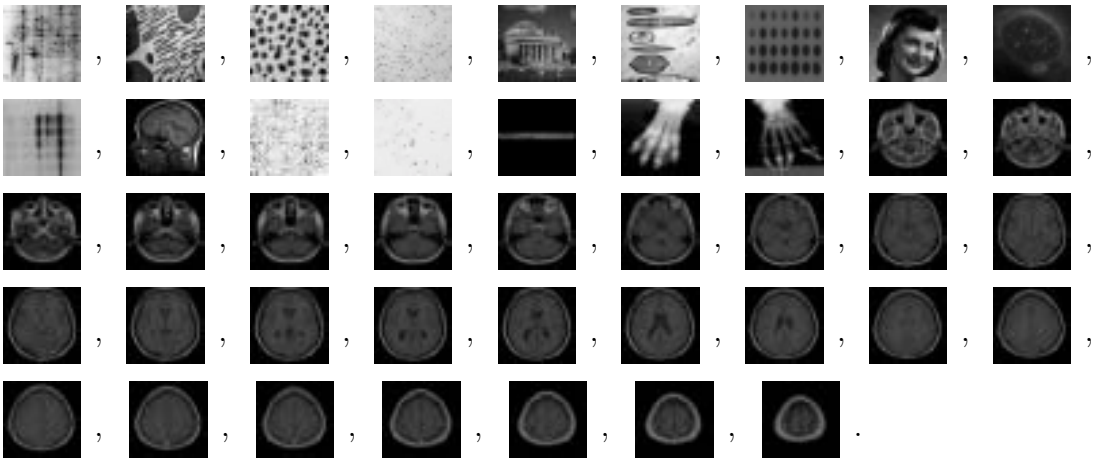
- *Grald Deshais*, Department of Materials Science & Metallurgy, University of Cambridge:



- *Ahmed Desoky*, *www.spd.louisville.edu/ ahdeso01*:

- *www.erc.msstate.edu/ rjm*:

- National Institute of Mental Health, Research Services Branch, Image Database:

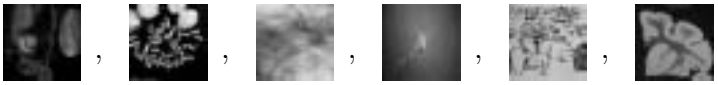


- Massachusetts Institute of Technology:

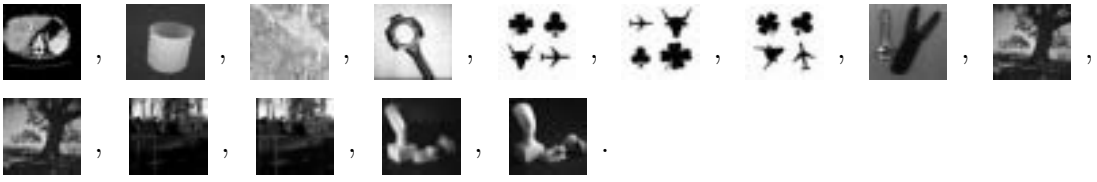
- *www.sys.uea.ac.uk*:

- *cs.haifa.ac.il/hagit*:

- *ipagwww.med.yale.edu/ staib*:



- McGraw–Hill:



- Computational Vision at Caltech image database, *www.vision.caltech.edu/*:

, , .

- CMU–MIT Face Database:

, , , , , , , , ,
, , , , , , , , ,
, , , , , , , , ,
, , , , , , , , ,
, , , , , , , , ,
, , , , , , , , ,
, , , , , , , , ,
, , , , , , , , ,
, , , , , , , , ,
.

- CMU Profile Face Database:

, , , , , , , , ,
, , , .

- *Alan W. Partin*, M.D., Ph.D., Johns Hopkins University School of Medicine:

.

- *Steve Decker* and *Shujaat Nadeem*, MIT:

  .

- Massachusetts Executive Office of Environmental Affairs, MassGIS:

  , .

- *Michael Myers*:

  .

- *J. C. Russ* [307]:

  , , , , , , , , ,
  , , , , , , , , ,
  , , .

- *www.mathworks.com*:

  , , , , , , , , ,
  , , , , , , , , ,
  , , , , , , , , ,
  , , , .

- *www.freeimages.co.uk*:

.

- NASA, *www.nasa.gov*:

, , , , , , , , ,
, , , .

- Vanderbilt University Medical Center, Department of Radiology and Radiological Sciences:

, , .

- Univ. of Michigan Medical School, Division of Anatomical Sciences:

, , .

- Lixi, Inc., *www.lixi.com*:

, .

- Florida State University, National High Magnetic Field Laboratory:

, , .

- *www.imageprocessingbook.com*:

, , , , , , , , ,
, , , , , , , , .

- Perceptics Corporation, *www.perceptics.com*:

.

- National Institute of Standards and Technology *www.nist.gov*:

.

- Siemens Medical Systems, Inc., Ultrasound Group:

, .

- McMaster University, Brockhouse Institute for Materials Research:

.

- General Electric, Medical Systems Division, *www.gemedicalsystems.com*:

.

- Australian National University, Research School of Biological Sciences:

.

- NTB electronische Gerate GmbH, Germany, *www.ntbxray.com*:

.

- X–TEK Systems, Ltd. *www.xtek.co.uk*:

.

Appendix C

Computation Cost

This appendix includes accurate evaluation of PCA–based Color Image Processing tools in terms of *floating point operations* (flop) needed to accomplish them. One flop refers to a floating operation like $a = b * c + d$, where all numbers are floating point numbers. Here, we assume the more complex operators of a^b, $\sin a$, $\cos a$, and truncation, $(y = [x])$, as a single flop too. Note that no special attempt in this section is made to optimize the operations.

Everywhere in this article, crisp variables are denoted as alphabetic or Latin characters like a and α, while fuzzy variables are indicated by a $\sim$ like $\tilde{a}$ and $\tilde{\alpha}$. Also, we assume that operations like vector transposition are accomplished in algorithm implementation.

The basic set on which algorithms perform is the set of n, d dimensional vectors $\vec{x}_i, i = 1, \cdots, n$. These vectors construct the set,

$$X = \left\{ \vec{x}_i \middle| 1 \leq i \leq n \right\}. \tag{C.0.1}$$

In fuzzy circumstances, we use the fuzzy set,

$$\tilde{X} = \Big\{(\vec{x}_i; p_i)\Big| 1 \leq i \leq n\Big\}. \tag{C.0.2}$$

For color image processing tools we perform a typical estimation by computing the demand of the algorithm on a 512×512 image. Also, when image sequence is considered the frame rate is assumed to be 30. To estimate algorithms' practical sped, we consider implementation on DSP processors like $DIOPSIS^{TM}740$ by *Atmel* [12] or $TMS320C6713$ by *Texas Instruments* [13] (came into market in 2001) which give the ability to process in 1 GFlops speed. As mentioned before, this section does not concern possible further optimizations. Table C.3 lists the computational cost of the operations discussed here.

C.1 Basic Operations

C.1.1 Vector Addition

Crisp

Addition of two vectors $\vec{x}_1$ and $\vec{x}_2$ gives,

$$\vec{y} = \vec{x}_1 + \vec{x}_2 \rightarrow y_i = x_{1i} + x_{2i}, 1 \leq i \leq d. \tag{C.1.3}$$

Hence, d flops are needed to accomplish a vector addition.

Fuzzy

Addition of the two fuzzy vectors $\tilde{\vec{x}}_1 = (\vec{x}_1; p_1)$ and $\tilde{\vec{x}}_2 = (\vec{x}_2; p_2)$ gives,

$$\vec{y} = \frac{p_1\vec{x}_1 + p_2\vec{x}_2}{p_1 + p_2}, \tag{C.1.4}$$

$$y_i = \frac{p_1}{p_1 + p_2}\Big(x_{1i} + \frac{p_2}{p_1}x_{2i}\Big), 1 \leq i \leq d, \tag{C.1.5}$$

$$y_i = \frac{1}{1 + p}\Big(x_{1i} + px_{2i}\Big), p = \frac{p_2}{p_1}, 1 \leq i \leq d. \tag{C.1.6}$$

Hence, $2d+3$ flops are needed to accomplish a fuzzy vector addition. Note that when $p_1 + p_2$ equals unity, the number of flops reduces to $2d$.

Here, we also consider the special case of fuzzy vector addition when $p_2 = 1$,

$$\vec{y} = p_1\vec{x}_1 + \vec{x}_2 \rightarrow y_i = p_1x_{1i} + x_{2i}, 1 \leq i \leq d, \tag{C.1.7}$$

which elapses d flops.

C.1.2 Vector Multiplication

The result of multiplying $\vec{x}$ with r is defined as,

$$\vec{y} = r\vec{x} \rightarrow y_i = rx_i, 1 \leq i \leq n, \tag{C.1.8}$$

elapsing d flops.

C.1.3 Vector Inner Product

Crisp

The inner product of $\vec{x}_1$ and $\vec{x}_2$ is defined as,

$$y = < \vec{x}_1, \vec{x}_2 > = \vec{x}_1^T\vec{x}_2 = \sum_{i=1}^{d} x_{1i}x_{2i}. \tag{C.1.9}$$

We propose the series,

$$y_i = \begin{cases} x_{1i}x_{2i} & i = 1 \\ y_{i-1} + x_{1i}x_{2i} & 1 < i \leq n \end{cases}. \tag{C.1.10}$$

Now, $y = y_n$. As such, the whole process demands d flops.

Fuzzy

The fuzzy inner product of $\vec{x}_1$ and $\vec{x}_2$ with weight vector of $\vec{p}$ is defined as,

$$y =< \vec{x}_1, \vec{x}_2 >_{\vec{p}} = \frac{\sum_{i=1}^{d} p_i x_{1i} x_{2i}}{\sum_{i=1}^{d} p_i}. \tag{C.1.11}$$

We propose the series,

$$y_i = \begin{cases} p_i x_{1i} x_{2i} & i = 1 \\ y_{i-1} + p_i x_{1i} x_{2i} & 1 < i \leq n \end{cases}. \tag{C.1.12}$$

Now,

$$y = \frac{y_n}{\sum_{i=1}^{d} p_i}. \tag{C.1.13}$$

In this way, the whole process demands $2d + d = 3d$ flops. Note that if $\sum_{i=1}^{d} p_i$ is available (*e.g.*, when performing a set of inner products with identical weights), the number of flops reduces to $2d + 1$.

C.1.4 Matrix Multiplication

Vector by Matrix Multiplication

Assume multiplying the d–D vector $\vec{x}$ by the $n \times d$ matrix A, ($\vec{y} = A\vec{x}$). We have,

$$y_i = \sum_{j=1}^{n} A_{ij} x_j =< \vec{A}_i, \vec{x} >, 1 \leq i \leq d, \tag{C.1.14}$$

where $\vec{A}_i$ denotes the i–th row of A. Thus, the whole process contains n times inner product computation between d–D vectors (see Section C.1.3). The total number of flops gets nd.

Matrix by Matrix Multiplication

Assume multiplying the $m \times p$ matrix C by the $n \times m$ matrix B. The result is the $n \times p$ matrix A, $A = BC$. We have,

$$A_{ij} = \sum_{k=1}^{m} B_{ik} C_{kj} =< \vec{B}_i, \vec{C}_j >, 1 \leq i \leq n, 1 \leq j \leq p, \tag{C.1.15}$$

where $\vec{B}_i$ and $\vec{C}_j$ denote the i–th row of B and the j–th column of C, respectively. Thus, the whole process contains np times inner product computation between m–D vectors (see Section C.1.3). The total number of flops gets nmp.

C.1.5 Convolution

Assume applying the convolution kernel M which is a $w \times h$ array of real values on the $W \times H$ grayscale image I. The operation, in each pixel, is modelled by an inner product between to wh–D vectors (see Section C.1.3). Thus, applying the convolution kernel elapses $WHwh$ flops.

C.1.6 Butterworth Fuzzy Membership Function

The color transfer process discussed in Section 3.9.2 utilizes the butterworth fuzzification function defined as,

$$B(x) = \left(1 + \left(\frac{x}{\tau}\right)^{2N}\right)^{-\frac{1}{2}}, \tag{C.1.16}$$

which needs 4 flops.

C.2 Statistics

C.2.1 Expectation Vector

Crisp

The expectation vector of X is computed as, $\vec{\eta}_X = \frac{1}{n}\sum_{i=1}^{n}\vec{x}_i$. We propose the series,

$$\vec{\eta}_{X,i} = \begin{cases} \vec{x}_1 & i = 1 \\ \vec{\eta}_{X,i-1} + \vec{x}_i & 1 < i \leq n \\ \frac{1}{n}\vec{\eta}_{X,i-1} & i = n+1 \end{cases} . \tag{C.2.17}$$

Now, $\vec{\eta}_X = \vec{\eta}_{X,n+1}$. Thus, computation of the expectation vector includes $n-1$ vector additions and a single vector multiplication. As such, the total needed flops equals $(n-1)d + d = nd$.

Fuzzy

Like the crisp expectation vector, the expectation vector of $\tilde{X}$ is computed as,

$$\vec{\eta}_{\tilde{X}} = \frac{\sum_{i=1}^{n} p_i\vec{x}_i}{\sum_{i=1}^{n} p_i}. \tag{C.2.18}$$

We propose to compute the series,

$$\vec{\eta}_{\tilde{X},i} = \begin{cases} p_1\vec{x}_1 & i = 1 \\ \vec{\eta}_{\tilde{X},i-1} + p_i\vec{x}_i & 1 < i \leq n \\ \frac{1}{\sum_{i=1}^{n} p_i}\vec{\eta}_{\tilde{X},i-1} & i = n+1 \end{cases} . \tag{C.2.19}$$

Now, $\eta_{\tilde{X}} = \vec{\eta}_{\tilde{X},n+1}$. Thus, computation of the fuzzy expectation vector includes $n-1$ fuzzy vector additions, two vector multiplications, and n extra flops (for computing $1/\sum_{i=1}^{n} p_i$). Hence, the total needed flops equals $(n-1)d + 2d + n = n(d+1) + d$.

C.2.2 Standard Deviation

Crisp

The standard deviation of the crisp set X of 1–D real values is defined as,

$$\sigma_X^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \eta_X)^2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)^2, \tag{C.2.20}$$

which after n flops of computing $X_2 = \{x_1^2, \cdots, x_n^2\}$, turns into two times computation of expectation vectors of 1–D arrays (see Section C.2.1) plus two extra flops for finishing the computation. Hence, the total number of flops equals $t = n + 2n + 2 = 3n + 2$.

Fuzzy

The standard deviation of the fuzzy set $\tilde{X}$ of 1–D real values is defined as,

$$\sigma_{\tilde{X}}^2 = \frac{\sum_{i=1}^{n} p_i(x_i - \eta_X)^2}{\sum_{i=1}^{n} p_i}, \tag{C.2.21}$$

$$\sigma_{\tilde{X}}^2 = \frac{\sum_{i=1}^{n} p_i x_i^2}{\sum_{i=1}^{n} p_i} - \Big(\frac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} p_i}\Big)^2, \tag{C.2.22}$$

which after n flops of computing,

$$\tilde{X}_2 = \{(x_1^2; p_1), \cdots, (x_n^2; p_n)\}, \tag{C.2.23}$$

turns into two times computation of fuzzy expectation vectors of 1–D arrays (see Section C.2.1) plus two extra flops for finishing the computation. Note that as the set of fuzzy membership values in the two sets are identical, we can save n flops. In this way, the total number of flops equals $4n + 4$ flops.

C.2.3 Maximum and Minimum

For the set X with N_X members, maximum is computed using the series,

$$y_i = \begin{cases} x_i & i = 1 \\ \max(x_i, y_{i-1}) & otherwise. \end{cases}, \tag{C.2.24}$$

which elapses $N_X - 1$ flops, where N_X denotes the number of members of X. In the same way, minimum elapses the same number of flops.

C.2.4 Histogram (PDF) Estimation

Crisp

Assume estimating the b–bin histogram of X. After creating a b–D vector h filled with zeros, we propose the computation of the following series,

$$h_i(X) = h_{i-1}(X) + \vec{e}\left(\left[b\frac{x_i - X_{min}}{X_{max} - X_{min}}\right]\right), \tag{C.2.25}$$

$$\begin{aligned} \vec{h}_i(X) &= h_{i-1}(X) + \\ &\vec{e}\left(\left[\frac{b}{X_{max} - X_{min}}x_i - \frac{bX_{min}}{X_{max} - X_{min}}\right]\right) \\ &= h_{i-1}(X) + \vec{e}\left(\left[b_1 x_i - b_2\right]\right), \end{aligned} \tag{C.2.26}$$

where X_{min} and X_{max} are the smallest and the largest members of X and,

$$b_1 = \frac{b}{X_{max} - X_{min}}, b_2 = \frac{bX_{min}}{X_{max} - X_{min}}. \tag{C.2.27}$$

Also,

$$\vec{e}(i)_j = \begin{cases} 1 & i = j \\ 0 & otherwise. \end{cases} \tag{C.2.28}$$

Now, PDF of X equals $pdf(X) = \frac{1}{n}\vec{h}$. Thus, spending $2(N_X - 1)$ flops to find the maximum and the minimum of X (see Section C.2.3) and 3 flops for computing b_1 and b_2, 3 flops for each member of X plus an extra flop are needed to complete the histogram. Hence, the total number of flops gets $5N_X + 2$ flops.

Fuzzy

In fuzzy domain the only change with discussion in Section C.2.4 is,

$$h_i(\tilde{X}) = h_{i-1}(\tilde{X}) + p_i\vec{e}\left(\left[b\frac{x_i - X_{min}}{X_{max} - X_{min}}\right]\right), \tag{C.2.29}$$

while we have,

$$pdf(X) = \frac{1}{\sum_{i=1}^{N_X} p_i}\vec{h}, \tag{C.2.30}$$

While the additional p_i term in (C.2.29) adds no extra flop, the $1/\sum_{i=1}^{N_X} p_i$ term in (C.2.30) adds N_X flops. Thus, the total number of flops equals $6N_X + 2$ flops.

C.2.5 CDF Estimation

Computing the CDF is performed using,

$$H_i = \begin{cases} 1 & i = b \\ H_{i+1} - h_i & 1 \leq i < n \end{cases}, \tag{C.2.31}$$

where h denotes the PDF (see Section C.2.4). Assuming that the proper PDF is available, the following operation elapses $N_X - 1$ operations. This makes the number of flops needed for estimating the CDF equal to $6N_x + 1$ flops.

C.2.6 Covariance Matrix

Crisp

The covariance matrix of X is defined as,

$$C_X = \frac{1}{n}\sum_{i=1}^{n}(\vec{x}_i - \vec{\eta}_X)(\vec{x}_i - \vec{\eta}_X)^T, \tag{C.2.32}$$

$$C_{Xij} = \frac{1}{n}\sum_{k=1}^{n}(x_{ki} - \eta_{Xi})(x_{kj} - \eta_{Xj}), 1 \leq i, j \leq d, \tag{C.2.33}$$

$$C_{Xij} = \frac{1}{n}\sum_{k=1}^{n} x_{ki}x_{kj} - \eta_{Xi}\eta_{Xj}, 1 \leq i, j \leq d, \tag{C.2.34}$$

$$C_{Xij} = \frac{1}{n} < \tau_i, \tau_j > -\eta_{Xi}\eta_{Xj}, 1 \leq i, j \leq d, \tag{C.2.35}$$

where,

$$\tau_i = \left(x_{1i}, \cdots, x_{ni}\right)^T, 1 \leq i \leq d. \tag{C.2.36}$$

Having in mind that the covariance matrix is symmetrical ($C_{Xij} = C_{Xji}$), we only have to compute $d(d-1)/2$ elements of C_X. Assuming that $\vec{\eta}_X$ is available (see Section C.2.1), computation of each element needs an inner product between two n–D vectors plus two extra flops. Thus, the whole operation needs $\frac{d(d-1)}{2}(n+2)$ flops.

Fuzzy

The covariance matrix of X is defined as,

$$C_{\tilde{X}} = \frac{\sum_{i=1}^{n} p_k(\vec{x}_i - \vec{\eta}_X)(\vec{x}_i - \vec{\eta}_X)^T}{\sum_{k=1}^{n} p_k}, \tag{C.2.37}$$

$$C_{\tilde{X}ij} = \sum_{k=1}^{n} p_k(x_{ki} - \eta_{Xi})(x_{kj} - \eta_{Xj}), 1 \leq i, j \leq d, \tag{C.2.38}$$

$$C_{\tilde{X}ij} = \frac{\sum_{k=1}^{n} p_k x_{ki} x_{kj}}{\sum_{k=1}^{n} p_k} - \eta_{Xi}\eta_{Xj}, 1 \leq i, j \leq d, \tag{C.2.39}$$

$$C_{\tilde{X}ij} =< \tau_i, \tau_j >_p - \eta_{Xi}\eta_{Xj}, 1 \leq i, j \leq d, \tag{C.2.40}$$

where,

$$\tau_i = \Big(x_{1i}, \cdots, x_{ni}\Big)^T, 1 \leq i \leq d, \tag{C.2.41}$$

$$p = \Big(p_1, \cdots, p_n\Big)^T, 1 \leq i \leq d. \tag{C.2.42}$$

Having in mind that the fuzzy covariance matrix is symmetrical ($C_{\tilde{X}ij} = C_{\tilde{X}ji}$), we only have to compute $d(d-1)/2$ elements of $C_{\tilde{X}}$. Assuming that $\vec{\eta}_{\tilde{X}}$ is available (see Section C.2.1), computation of each element needs a fuzzy inner product between two n–D vectors plus one extra flops. Note that the fuzzy inner products are performed on the same set of weights. Thus, the whole operation needs,

$$\frac{d(d-1)}{2}\big((2n+1) + n + 1\big) = \frac{d(d-1)}{2}(3n+2), \tag{C.2.43}$$

flops.

C.2.7 SVD (PCA)

The theory of *singular value decomposition* (SVD) states that any matrix X can be decomposed as $X = VDU^T$. Assuming that X is a $n \times m$ matrix with rank d, D will

be a diagonal $d \times d$ matrix while V and U are $n \times d$ and $d \times m$ matrices, respectively. When X is a symmetrical $d \times d$ matrix V, D, and U are all $d \times d$ matrices with U and V being orthogonal satisfying $U = V^T$.

Assume the set X of d–D vectors is given. Also, assume computing the covariance matrix of X as C_X (see Section C.2.6). Now, C_X is decomposed as the diagonal form of $C_X = V_X D_X V_X^T$. The principal components of $\vec{x}_i$ in this framework are given by $\vec{y}_i = V_X^T(\vec{x}_i - \vec{\eta}_X)$. As such, the components of $\vec{y}_i$ are ordered by their importance. This is the classic theory of *principal component analysis* (PCA) [183].

Computation of V and U elapse $2mn^2$ flops [151]. Thus, when the covariance matrix and the expectation vector are available, computation of PCA matrix elapses $2d^3$ flops (note that $d \ll m, n$).

C.3 PCA–Based Color Image Processing

C.3.1 LPRE

The LPRE distance of vector $\vec{c}$ to cluster r is defined as:

$$\tau_X(\vec{x}) = \|\vec{v}_X^T(\vec{x} - \vec{\eta}_X)\vec{v}_X - (\vec{x} - \vec{\eta}_X)\|_2^2, \tag{C.3.44}$$

where $\vec{v}_X$ denotes the direction of the first principal component and $\|\vec{x}\|_2^2$ is the squared 2–norm of $\vec{x}$. Computation of $\tau_X(\vec{x})$ includes computing two vector additions, two inner products, and one vector multiplication. Hence the total number of operations needed for a single LPRE gets $2d + 2d + d = 5d$.

C.3.2 Normalized LPRE and LPRE–based Homogeneity Criterion

The normalization scheme for LPRE uses the following stochastic margin as the normalization factor:

$$\|f\|_{X,p} = \arg_e \Big\{ P_{\vec{x}\in X}\{f(\vec{x}) \leq e\} \geq p \Big\}, \tag{C.3.45}$$

where, p is the inclusion percentage and $P_{\vec{x}\in X}\{f(\vec{x}) \leq e\}$ denotes the probability of x being less than or equal to e. Therfore, the normalized LPRE is defined as,

$$\tilde{\tau}_{X,p}(\vec{x}) = \frac{\tau_X(\vec{x})}{\|\tau_X\|_{X,p}}. \tag{C.3.46}$$

It is also proved that the denominator term $\|\tau_X\|_{X,p}$ serves as a proper homogeneity factor for swatches in color images of the nature.

Assuming that an arbitrary function $f : R^d \rightarrow R$ elapses α flops for each single input, we investigate the total flops needed to compute $\|f\|_{X,p}$. We first have to construct the following set,

$$f(X) = \{f(x)|x \in X\}, \tag{C.3.47}$$

which elapses αN_X flops. Here, N_X denotes the number of members of X. Then the process have to spend $6N_X + 1$ flops for computing the CDF of $f(X)$ (see Section C.2.5). After that the process should elapse at most N_X flops for finding the nearest value in $cdf(X)$ to p. Hence, the total number of flops needed to compute

$\|f\|_{X,p}$ equals $(\alpha + 7)N_X + 1$ flops. For the case of computing $\|\tau_X\|_{X,p}$, the α equals $5d$ (see Section C.3.1), resulting in a total of $(5d + 7)N_X + 1$ flops.

When $\vec{v}_X$, $\vec{\eta}_X$, and $\|\tau_X\|_{X,p}$ are available, computing the normalized LPRE elapsed only one more flop compared to the LPRE.

C.3.3 Single–Swatch Recoloring

Assume applying the single–swatch recoloring discussed in Section 3.9.2 on I to produce J,

$$\vec{J}_{ij} = V_{r_2} V_{r_1}^T (\vec{I}_{ij} - \vec{\eta}_{r_1}) + \vec{\eta}_{r_2}, \tag{C.3.48}$$

where r_1 and r_2 are two $w \times h$ swatches. The process needs to compute the PCA matrix and expectation vector of r_1 and r_2 swatches which takes,

$$t_1 = 2\big(2d^3 + \frac{d(d-1)}{2}(wh + 2) + whd\big), \tag{C.3.49}$$

flops. Having in mind that d (dimension of vectors) is three, we have $t_1 = 12wh + 120$ flops.

Assume rewriting (C.3.48) as,

$$\vec{J}_{ij} = V_{r_2} V_{r_1}^T \vec{I}_{ij} + \vec{\eta}_{r_2} - V_{r_2} V_{r_1}^T \vec{\eta}_{r_1}. \tag{C.3.50}$$

Hence, the process first needs to compute one matrix by matrix multiplication, one vector by matrix multiplication, and one vector addition for each pair of swatches. As such, $V = V_{r_2} V_{r_1}^T$ and $\vec{\eta} = \vec{\eta}_{r_2} - V_{r_2} V_{r_1}^T \vec{\eta}_{r_1}$ are computed. Now, (C.3.48) changes to $\vec{J}_{ij} = V\vec{I}_{ij} + \vec{\eta}$. Computation of V and $\vec{\eta}$ takes $d^3 + d^2 + d = 39$ flops.

Then, for each pixel, the process have to compute one vector additions and one vector by matrix multiplications. This makes the number of flops for processing each pixel equal to $3d + d = 4d = 12$ flops ($d = 3$).

Thus, the total number of flops for a complete single–swatch recoloring equals $12WH + 12wh + 159$ flops, which depends linearly on the image size.

C.3.4 Color Space Conversion

Here, we investigate the computational cost of converting a single pixel from RGB representation to other color spaces (see Table C.1 and Table C.2). The investigation includes the computational demand of each color space in its original form plus the cost of normalizing it to the $[0, 255]$ interval. As an overall estimate, 7.5 ± 4.8 flops are needed to convert a pixel from RGB to another color space. Using the normalized representations increases this figure to 8.7 ± 5.3. In all cases related to computing the $CIE - f$ function, its computational cost is assumed to 2 flops.

Table C.1: Computational cost of converting one pixel to a color space. All values are in terms of flops.

Color Space	**Representation**						
	x_1	x_2	x_3	x_1x_2	x_2x_3	x_3x_1	$x_1x_2x_3$
$HMMD$	9	2	2	11	4	11	13
$I_1I_2I_3$	3	2	3	5	5	6	8
HSI	9	4	3	13	7	12	16
$CIE-XYZ$	3	3	3	6	6	6	9
HSV	9	6	2	15	8	11	17
YC_bC_r	3	2	2	5	4	5	7
$CIE-Lab$	3	6	6	9	12	9	15
YIQ	3	3	3	6	6	6	9
$CIE-LHC$	4	10	3	14	13	7	17
YUV	3	3	3	6	6	6	9
$CIE-Luv$	7	11	8	18	19	15	26
IR_gB_y	2	3	4	5	7	6	9
$IR_gB_y^+$	3	4	5	7	9	8	12
$Nrgb$	3	1	1	4	2	4	5
YUV°	3	6	5	9	11	8	14
YIQ°	3	6	5	9	11	8	14
$YC_bC_r^\circ$	3	5	4	8	9	7	12
HSI^c	11	6	3	17	9	14	20
HSV^c	11	8	2	19	10	13	21
RGB^r	2	2	2	4	4	4	6
TSL	5	9	3	14	12	8	17
Average	4.9	4.9	3.4	9.7	8.3	8.3	13.1
Std. Dev.	3.1	2.8	1.6	4.8	3.9	3.2	5.4
Average	4.4			8.8			13.1
Std. Dev.	2.6			4.0			5.4

Table C.2: Computational cost of converting one pixel to a color space in its $[0, 255]$ normalized formulation. All values are in terms of flops.

Color Space	**Representation**						
	x_1	x_2	x_3	x_1x_2	x_2x_3	x_3x_1	$x_1x_2x_3$
$HMMD$	10	2	2	12	4	12	14
$I_1I_2I_3$	3	3	4	6	7	7	10
HSI	10	5	3	15	8	13	18
$CIE-XYZ$	3	3	4	6	7	7	10
HSV	10	7	2	17	9	12	19
YC_bC_r	3	3	3	6	6	6	9
$CIE-Lab$	4	7	7	11	14	11	18
YIQ	3	4	4	7	8	7	11
$CIE-LHC$	5	11	4	16	15	9	20
YUV	3	4	4	7	8	7	11
$CIE-Luv$	8	12	9	20	21	17	29
IR_gB_y	3	4	5	7	9	8	12
$IR_gB_y^+$	4	5	6	9	11	10	15
$Nrgb$	4	2	2	6	4	6	8
YUV°	3	7	6	10	13	9	16
YIQ°	3	7	6	10	13	9	16
$YC_bC_r^\circ$	3	6	5	9	11	8	14
HSI^c	12	7	3	19	10	15	22
HSV^c	12	9	2	21	11	14	23
RGB^r	2	2	2	4	4	4	6
TSL	6	10	3	16	13	9	19
Average	5.4	5.7	4.1	11.1	9.8	9.5	15.2
Std. Dev.	3.4	3.0	1.9	5.3	4.2	3.3	5.7
Average	5.1			10.2			15.2
Std. Dev.	2.9			4.3			5.7

Table C.3: Computational cost of basic operations in terms of flops. When nothing else is restricted, all vectors are d–D, all matrices are $d \times d$, and X and $\tilde{X}$ are sets with n members. I is always a $H \times W$ grayscale image. Number of flops indicated by $\star$ are those for applying the same operation on a bunch of values. [**C**: Crisp, **F**:Fuzzy, **V**: Conventional, **N**: Normalized.]

Operation		**Formulation**	**Cost**
Vector Addition	**C**	$\vec{y} = \vec{x}_1 + \vec{x}_2$	d
	F	$\vec{y} = \frac{p_1\vec{x}_1 + p_2\vec{x}_2}{p_1+p_2}$	$2d+3$
		$\vec{y} = \vec{x}_1 + p\vec{x}_2$	d
Vector Multiplication		$\vec{y} = r\vec{x}$	d
Inner Product	**C**	$y = \sum_{i=1}^{d} x_{1i}x_{2i}$	d
	F	$y = \frac{1}{\sum_{i=1}^{d} p_i} \sum_{i=1}^{d} p_i x_{1i} x_{2i}$	$3d$ or $2d+1^\star$
Matrix Multiplication		$\vec{y}_{n\times 1} = A_{n\times d}\vec{x}_{d\times 1}$	nd
		$A_{n\times p} = B_{n\times m}C_{m\times p}$	nmp
Convolution		$J = M_{h\times w} * I$	$WHwh$
Butterworth		$B(x) = \left(1 + \left(\frac{x}{\tau}\right)^{2N}\right)^{-\frac{1}{2}}$	4
Expectation Vector	**C**	$\vec{\eta}_X = \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i$	nd
	F	$\vec{\eta}_{\tilde{X}} = \frac{\sum_{i=1}^{n} p_i\vec{x}_i}{\sum_{i=1}^{n} p_i}$.	$n(d+1)+d$
Standard Deviation	**C**	$\sigma_X^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \eta_X)^2$	$3n+2$
	F	$\sigma_{\tilde{X}}^2 = \frac{\sum_{i=1}^{n} p_i(x_i - \eta_{\tilde{X}})^2}{\sum_{i=1}^{n} p_i}$	$4n+4$
Maximum		$y = \max_{x\in X}\{x\}$	$n-1$
Minimum		$y = \min_{x\in X}\{x\}$	$n-1$
PDF	**C**	$h_X(x)$	$5n+2$
	F	$h_{\tilde{X}}(x)$	$6n+2$
CDF		$H_X(x)$, $H_{\tilde{X}}(x)$	$6n+1$
Covariance Matrix	**C**	$C_X = \sum_{i=1}^{n}(\vec{x}_i - \vec{\eta}_X)(\vec{x}_i - \vec{\eta}_X)^T$	$\frac{d(d-1)}{2}(n+2)$
	F	$C_X = \frac{\sum_{i=1}^{n} p_i(\vec{x}_i - \vec{\eta}_X)(\vec{x}_i - \vec{\eta}_X)^T}{\sum_{i=1}^{n} p_i}$	$\frac{d(d-1)}{2}(3n+2)$
PCA Matrix		$V_X : C_X = V_X D_X V_X^T$	$2d^3$
LPRE		$\tau_X(\vec{x}) = \|\vec{v}_X^T(\vec{x} - \vec{\eta}_X)\vec{v}_X - (\vec{x} - \vec{\eta}_X)\|_2^2$	$5d$
$\tilde{\tau}_{X,p}(\vec{x})$		$\tilde{\tau}_{X,p}(\vec{x}) = \frac{\tau_X(\vec{x})}{\|\tau_X\|_{X,p}}$	$5d+1$
$\|\tau_X\|_{X,p}$		$\|\tau_X\|_{X,p}$	$(5d+7)n+1$
Recoloring		$\vec{J}_{ij} = V_{r_2}V_{r_1}^T(\vec{I}_{ij} - \vec{\eta}_{r_1}) + \vec{\eta}_{r_2}$	$12WH+$ $12wh+159$
$CIE-f$		$y = f(x)$	2
Color Space Conversion	**V**	$\vec{y} = F(\vec{x})$	7.5 ± 4.8
	N	$\vec{y} = F_N(\vec{x})$	8.7 ± 5.3

Appendix D

Multi-Layer Representation of Grayscale Images

D.1 Introduction

Thresholding is one of the most fundamental primitives of grayscale image processing. Accordingly, different approaches for *threshold selection* already exist in the literature [377, 312]. Using a priori knowledge about the share of each class in the image [121] or finding the deepest valley in the histogram [291] are two simple examples from the sixties. More recent approaches incorporate the edge information to the threshold–decider [378] and fit parametric curves to the histogram [279].

There are a few approaches that concern multilevel thresholding. As the generalization of single–level thresholding, the multi-level thresholding approaches work on finding a set of thresholds for converting an image into an index array. Two representative examples of these approaches are [355, 280].

D.2 Proposed Method

Assume that n realizations of the arbitrary random variable X are given as $S_X = \{x_1, \cdots, x_n\}$, $x_i \in R$. A multi–layer segmentation of S_X is a set of binary planes and corresponding multipliers. These parameters should be selected in a way that they give a proper approximation of the original signal. Below, we will discuss this definition in full details.

Lets work on the single–layer segmentation. For the sequence S_X, the single layer segmentation is a same–sized binary sequence $G_{1X} = [g_{11}, g_{12}, \cdots, g_{1n}]$ and two real values r_1 and s_1. Where, r_1 and s_1 are two real values. As such, the first approximation of x_i is denoted by $\tilde{x}_{1i}$ and is defined as,

$$\tilde{x}_{1i} = g_{1i} r_1 + (1 - g_{1i}) s_1. \tag{D.2.1}$$

Equation (D.2.1) shows that in the first layer, the sequence is quantized into two levels. Each quantization level is then assigned by a single representative. The above single–layer segmentation should be performed in a way that results in the least possible distortion. Hence, to obtain proper values of r_1, s_1, and g_{1i}, one should minimize the objective function of,

$$\Delta_1 = \sum_{i=1}^{n} (x_i - \tilde{x}_{1i})^2 = \sum_{i=1}^{n} \big(x_i - g_{1i} r_1 - (1 - g_{1i}) s_1\big)^2. \tag{D.2.2}$$

Computing the derivative of Δ_1 in terms of r_1 gives,

$$\frac{\partial \Delta_1}{\partial r_1} = -2 \sum_{i=1}^{n} g_{1i} \big(x_i - g_{1i} r_1 - (1 - g_{1i}) s_1\big). \tag{D.2.3}$$

Using the fact that $g_{1i}(1 - g_{1i})$ is always zero and $g_{1i}^2 = g_{1i}$, we get,

$$r_1 = E_{g_{1i}=1}\{x_i\}. \tag{D.2.4}$$

Similarly, computing $\frac{\partial \Delta_1}{\partial s_1}$ gives,

$$s_1 = E_{g_{1i}=0}\{x_i\}. \tag{D.2.5}$$

Here, $E_{c_i}\{x_i\}$ denotes the expectation of x_i for all those i that satisfy c_i, where c_i is a boolean condition.

Note that the computation of r_1 and s_1, does not solve the problem. Because these values depend essentially on the layer definition (G_{1X}). To solve this recursion, select an arbitrary value of i. The share of i in Δ_1 is δ_{1i} defined as,

$$\delta_{1i} = \big(x_i - g_{1i}r_1 - (1 - g_{1i})s_1\big)^2. \tag{D.2.6}$$

Hence, g_{1i} should be selected in a way that δ_{1i} gets minimum. As $g_{1i} \in [0, 1]$ we have to only check whether,

$$g_{1i} = 1 \leftrightarrow (x_i - r_1)^2 \leq (x_i - s_1)^2. \tag{D.2.7}$$

or,

$$g_{1i} = 1 \leftrightarrow x_i \leq \frac{1}{2}(r_1 + s_1). \tag{D.2.8}$$

Assuming that $r_1 < s_1$. Substituting (D.2.8) in (D.2.4) and (D.2.5) gives,

$$E_{x_i \leq \frac{1}{2}(r_1+s_1)}\{x_i\} = r_1, \tag{D.2.9}$$

$$E_{x_i>\frac{1}{2}(r_1+s_1)}\{x_i\} = s_1, \tag{D.2.10}$$

which should be solved simultaneously. We combine (D.2.9) and (D.2.10) to have,

$$E_{x_i\leq\frac{1}{2}(r_1+s_1)}\{x_i\} + E_{x_i>\frac{1}{2}(r_1+s_1)}\{x_i\} = r_1 + s_1. \tag{D.2.11}$$

Now, substituting $\theta = \frac{1}{2}(r_1 + s_1)$ in (D.2.11), we have,

$$E_{x_i\leq\theta}\{x_i\} + E_{x_i>\theta}\{x_i\} = 2\theta. \tag{D.2.12}$$

Hence, we are seeking for the zero point of,

$$f(\theta) = \frac{1}{2}\Big(E_{x_i\leq\theta}\{x_i\} + E_{x_i>\theta}\{x_i\}\Big) - \theta. \tag{D.2.13}$$

Note that for an arbitrary sequence of realizations of a random variable we have,

$$\lim_{x\to\min\{X\}} E_{x_i\leq x}\{x_i\} = \min\{X\}, \tag{D.2.14}$$

and

$$\lim_{x\to\max\{X\}} E_{x_i>x}\{x_i\} = \max\{X\}. \tag{D.2.15}$$

Hence,

$$f\Big(\min\{X\}\Big) = \frac{1}{2}\Big(\bar{X} - \min\{X\}\Big) \geq 0. \tag{D.2.16}$$

Here, $\bar{X}$ denotes the expectation of the sequence of values of X. Also, we have,

$$f\Big(\max\{X\}\Big) = -\frac{1}{2}\Big(\max\{X\} - \bar{X}\Big) \leq 0. \tag{D.2.17}$$

Note that (D.2.16) and (D.2.17) turn into equalities when and only when all the values in the sequence X are identical. Neglecting this impractical situation which

results in $f(\theta) \equiv 0$, we have,

$$f\Big(\max\{X\}\Big) f\Big(\min\{X\}\Big) < 0. \tag{D.2.18}$$

Hence, working in the interval $[\min\{X\}, \max\{X\}]$, to which θ should be a member of, bi–section gives a proper approximation of the value of θ giving $f(\theta) \simeq 0$. Note that in (D.2.13), $f : R \to R$ is a continuous function.

Just to perform a brief review, the method of bi–section for finding a zero of the continuous function $f : R \to R$ in the interval $[a, b]$ given that $f(a)f(b) < 0$ is as follows; First, $c = \frac{1}{2}(a+b)$ is computed. If $f(c)$ is zero, the goal is reached. Otherwise, one and only one of the two inequalities $f(c)f(a) < 0$ and $f(c)f(b) < 0$ should hold. Now, the search is repeated in the interval $[a, c]$ or $[c, b]$. The algorithm stops when the length of the working interval is less than a preselected threshold. Here, we set the minimum interval length, which determines the precision of θ, equal to 2^{-r} multiplied by the original range of the sequence. A nominal value of r is 10 which results in a precision of approximately 0.1% of the interval length. The method of bi–section is proved to linearly converge, as the error in each stage is half the error in the last stage. Assuming that the length of the beginning search interval equals α, after j iterations, it would equal $2^{-j}\alpha$. Hence, using the minimum threshold of $2^{-r}\alpha$ for convergence, the algorithm will stop not after r iterations. With the algorithm computing $f(\theta)$ only for interval margins (one computation of f at each iteration), the total number of times the function f is computed equals $r + 2$. For the nominal value, this is only

twelve times computation of f.

Returning to the original problem, each times computation of the function f in our case, means performing n comparisons and n additions. Hence, the price of finding θ by the bi–section approach equals $2n(r+2)$ flops (6.3 Mega flops for a 512×512 image while r equals 10, elapsing 6ms on a one–Giga–flops processor). Here, we have assumed that the original image is used for computing θ. Though, the wise idea would be to use a down–sampled version. Now, assume that we have down–sampled the original signal by a factor of λ before feeding it to the procedure (which finds the optimum θ). It is clear that reducing the number of data realizations may only (rarely) decrease the number of iterations passed before convergence. Hence, after down–sampling the cost of finding the optimum θ equals, $\frac{2}{\lambda}n(r+2)$. As such, by increasing λ and decreasing r, one can decrease the computational cost, drastically. In Section D.3, we will perform a through investigation to obtain practical values for λ and r. We emphasize that this down–sampling scheme is not useful in the areas where the realizations of the random variable are not correlated in the spatial (or time) domain. It means that for a set of data points gathered from the values of a (computer generated) random sequence, performing the down–sampling will result in data–loss. Though, the reader should be aware that images are highly correlated in the spatial domain and hence act perfectly in the proposed method.

There is another interpretation for the process performed at the above. Note that

PSNR depends on the amount of distortion, computed in terms of the mean square difference between the original and the approximated signal. Hence, the one–layer segmentation produced here, is the *best* one–bit representation of the given image in the mean square sense.

Now that the mathematics is developed, we turn into the original problem of multi–layer segmentation. Having the sequence $X = [x_1, \cdots, x_n]$, the m–layer segmentation is the set of n binary planes $G_m = [g_{11}, \cdots, g_{1n}], \cdots, [g_{m1}, \cdots, g_{mn}]$ plus $2n$ real values of $R_m = [r_1, \cdots, r_m]$ and $S_m = [s_1, \cdots, s_m]$. We define,

$$\tilde{x}_{m,i} = \sum_{j=1}^{m} \Big(g_{j,i} r_j + (1 - g_{j,i}) s_j \Big), \tag{D.2.19}$$

as the m–layer representation of x_i. Note that (D.2.1) is a special case of (D.2.19) with $m = 1$. Now, the objective function for fitting $x_{k,i}$s into x_is changes to,

$$\Delta_m = \sum_{i=1}^{n} \Big(x_i - \sum_{j=1}^{m} \big(g_{j,i} r_j + (1 - g_{j,i}) s_j \big) \Big)^2. \tag{D.2.20}$$

We call the combination of G_m, R_m, and S_m as a proper m–layer segmentation of X if for all $1 \leq k \leq m$, G_k, R_k and S_k give a proper k–layer segmentation of X (note that here X denotes a sequence of values of a random variable). This recursive definition implies that for finding the $(m+1)$–layer representation of X, one should first find its m–layer representation. Rewriting (D.2.19) as,

$$\tilde{x}_{k,i} = x_{k-1,i} + g_{k,i} r_j + (1 - g_{k,i}) s_k, \tag{D.2.21}$$

and (D.2.20) as,

$$\Delta_k = \sum_{i=1}^{n} \Big((x - \tilde{x}_{k-1,i}) - g_{k,i} r_k - (1 - g_{k,i}) s_k \Big)^2, \tag{D.2.22}$$

proves that the k–th layer is essentially the one–layer segmentation of the error of the $k-1$–layer representation. From this it proceeds that the process of finding the m–layer segmentation of a sequence of n realization (call it X) of a random variable is as follows: find the one–layer representation of X and call it X_1. Set $X = X - X_1$ and find the one–layer representation of the new X. Call the result X_2. Set $X = X - X_2$ and proceed to m layers.

The computational cost of obtaining the m–layer representation of a sequence of n values using the down–sampling ratio of λ and the precision of 2^{-r} equals $m(\frac{2}{\lambda} n(r+2) + 2n)$. For nominal values of $r = 10$ and $\lambda = 200$, the multiplier $\frac{2}{\lambda}(r+2)$ is less than $\frac{1}{8}$. Hence, the computational cost of the proposed method almost equals $2nm$. Here, n is the number of data points and m is the number of layers. As such, the cost of the proposed method is linear both in terms of the number of data points and the number of layers. In an ordinary practice, we may need to compute the eight–layer representation of a 512×512 image. This operation takes 4.2 Mega flops, elapsing less than $5ms$ on a one–Giga–flop processor.

The functions *MSCPbest*, *MSCPvalue*, and *MultiSegment* are developed to perform this operation.

MSCPbest.m

```
function [v,fv,loop]=MSCPbest(I,precision,SHOW,lambda)
%Function best multi-segmentation cutting point computes the best point to
%cut the signal I. Refer to my thesis for mathematics!
%Usage: [v,fv,loop]=MSCPbest(I,precision,SHOW,lambda);
%Input: I:The signal.
%       precision:Precision.
%           >0: It is the real precision.
%           <0: 2^precision*(max(I(:))-min(I(:))) is the precision.
%       SHOW: Show the results? (default: 0).
%       lambda:Downsampling ratio (default: 1).
%Output:v:The best point.
%       fv:The cost of cutting at v.
%       loop:Number of iterations.
%Revision 1.2, Arash Abadpour, Spring 2005 (May 18, 2005).
```

Figure D.1: Implementation of the *MSCPbest* function.

MSCPvalue.m

```
function e=MSCPvalue(I,t)
%Function multi-segmentation cutting point value comute sthe cost of
%cutting I at t.
%Usage: e=MSCPvalue(I,t);
%Input: I:The signal.
%       t:Cutting point.
%Output:e:The cost.
%Revision 1.0, Arash Abadpour, Spring 2005 (May 14, 2005).
```

Figure D.2: Implementation of the *MSCPvalue* function.

MultiSegment.m

```
function [Js,Es,P,v]=MultiSegment(I,thr,precision)
%Function multisegment performs multi segmentation.
%Usage:[Js,Es,P,v]=MultiSegment(I,thr,precision);
%Inputs: I:The image.
%       thr:if positive: thr determines the PSNR threshold (default: 38).
%           if negative: -thr determines the number of steps to go.
%       precision: Precision of computing the best cutting point
%       (default: -10). Refer to MSCPbest for defnition.
%Outputs: Js: The planes.
%        Es:Successive approximations.
%        P:Successive values of PSNR.
%        v:Levels.
%Revision 1.0, Arash Abadpour, Spring 2005 (May 16, 2005).
```

Figure D.3: Implementation of the *MultiSegment* function.

D.3 Experimental Results

Figure D.4–a, b, and c show *Lena* image, its 50–bin PDF, and the values of $f(x)$, respectively. Here, the optimum θ is approximated as 104.7. Figure D.4–d shows the one–layer segmentation of *Lena* using $r_1 = 72.2$ and $s_1 = 150.8$. The PSNR is about $20dB$.

Figure D.5 examines 512×512, $8bpp$ *Peppers* image to find the acceptable values of the precision and λ (down–sampling ratio of the signal for reducing the cost of finding θ). Figure D.5–a shows that selecting the precision of 2^{-4} is an acceptable compromise. Note that larger values of $-log_2(precision)$ tend to unwanted decrease

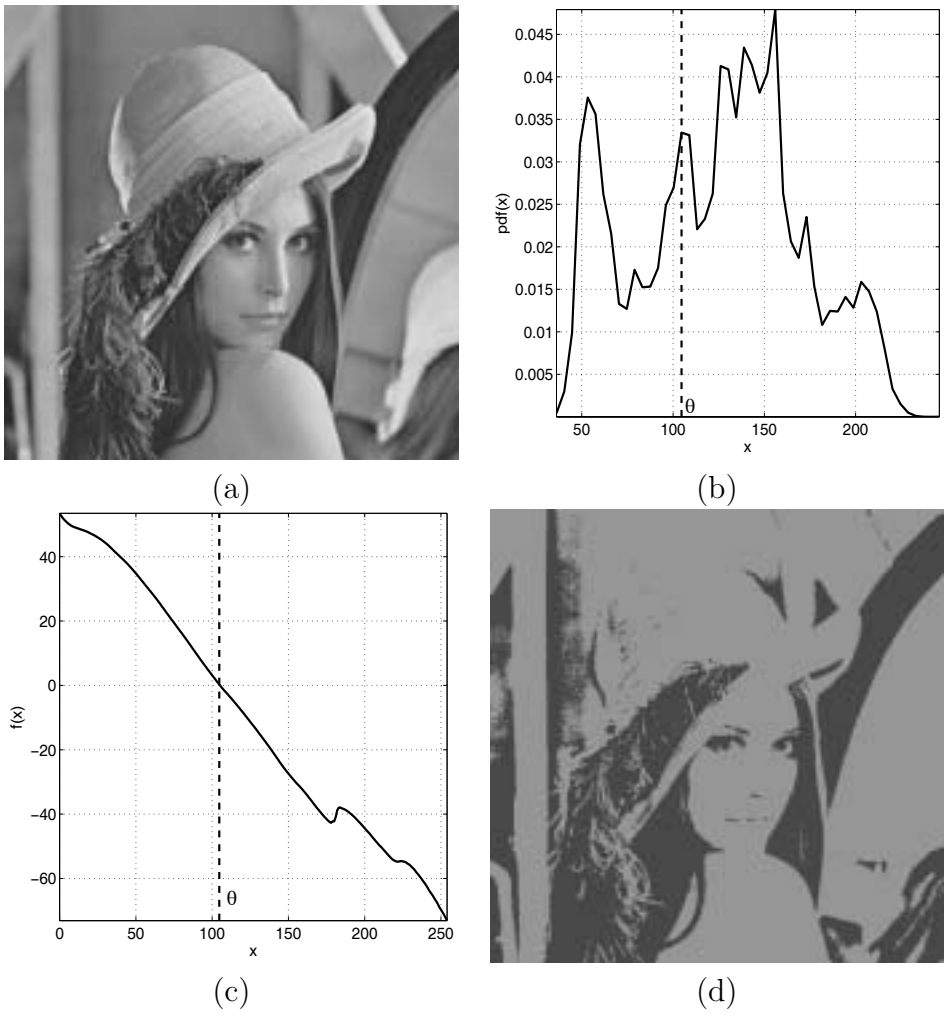
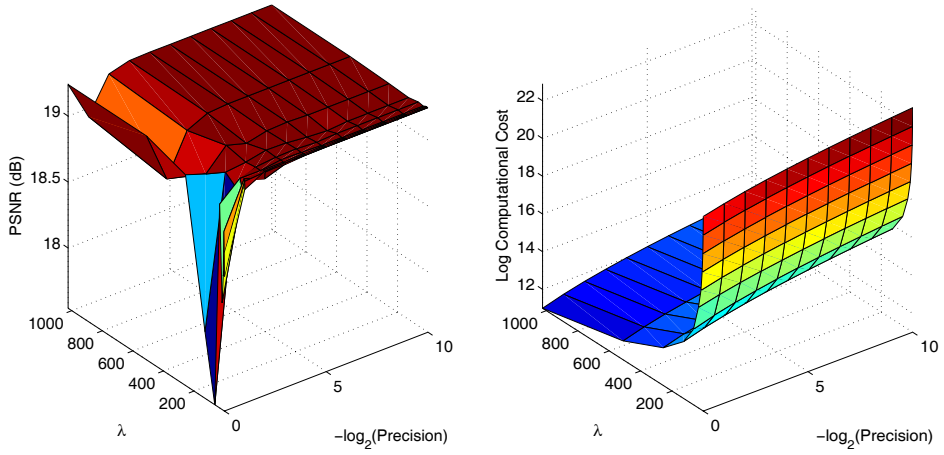


Figure D.4: (a) *Lena* image. (b) PDF of (a). (c) Values of the function f defined in (D.2.13). (d) Single–layer segmentation result (PSNR=19.8dB).



(a) (b)

Figure D.5: (a) PSNR of a single–layer segmentation of *Peppers* image with varying λ and r. (b) Logarithm of the computational cost for different values of λ and r.

of PSNR. Figure D.5–b shows that reducing the number of data points by a factor of 1000 has no great influence on the PSNR. Hence, according to the fact that having a large number of realizations, greatly declines the performance of finding θ in the coming parts of this thesis we down–sample he signal before feeding it to the routine which finds the optimum θ. Also, we let the routine to stop making θ more precise when it is working in an interval less than $\frac{1}{16}$ of the original interval. With this settings, the cost of computing the optimum θ declines to 3200 flops (about $3\mu s$ in a one–Giga–flop processor). Using non–optimized MATLAB code, it takes $28ms$ to perform the operation. From which, $22ms$ is elapsed on performing the down–sampling (on a PIV with clock frequency of $2871MHz$).

Figure D.6 shows the results of applying the proposed method on a 1–D signal.

This signal is actually a line of a 512×512 image. The value of SNR increases as $23dB$, $27dB$, $31dB$, $34dB$, and $39dB$ as the number of layers increases from one to five, respectively. We believe that this figure clearly depicts the scope of the proposed method.

Figure D.7 shows *unr730* image. The proposed method is applied on this image to produce the number of layers that give a proper representation with $PSNR \geq 40dB$. As such, seven layers have been produced (here, we have only shown the results of five layers, the two others are not visually recognizable). Note the results of adding more and more layers. While increasing the PSNR, adding new layers also results in an image with more details. Figure D.8 shows PSNR versus layer number for this image.

Here, we compare the proposed method with one of the best available single–layer thresholding approaches. As an outstanding approach, *Otsu* proposed a method for finding the optimal threshold of an image containing a bright object on a dark background [275]. The *Otsu* method divides a double–peak histogram into two compact classes which are well–separated. The above method differs from the proposed method in many major points. While the *Otsu* method only works for a specific class of images, it focuses on the intra– and inter–specifications of the classes. In contrast, the proposed method gives a proper estimation of any arbitrary image. Also, *Otsu*'s

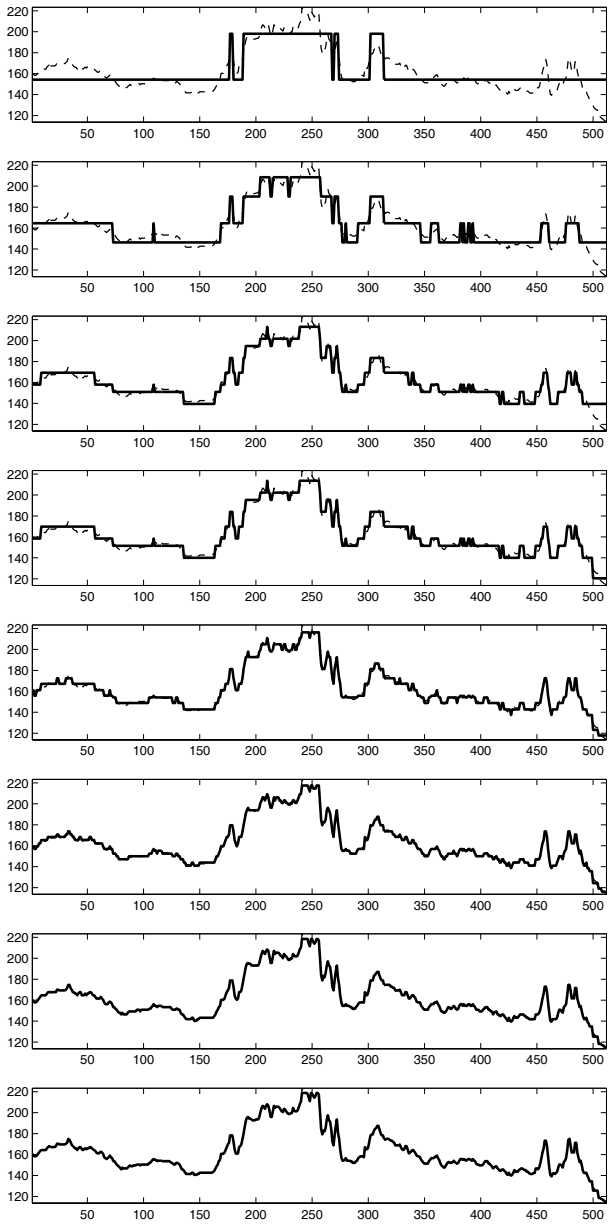
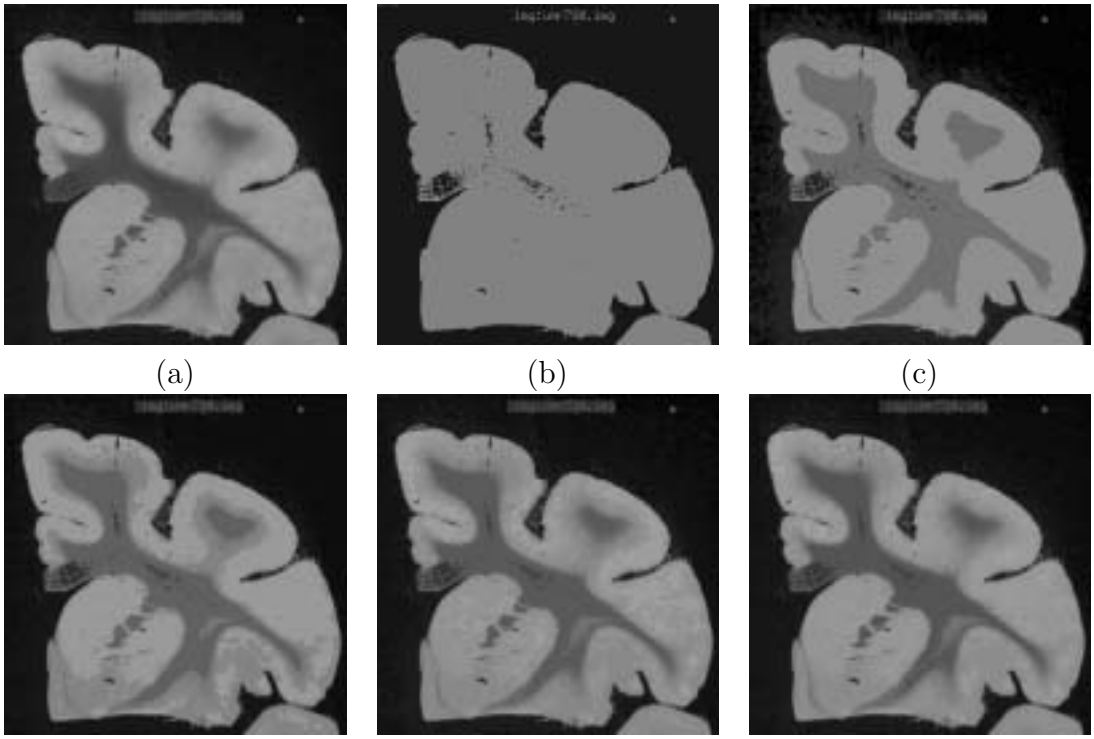


Figure D.6: Results of applying the proposed method on a 1–D signal.



(a) (b) (c)

(d) (e) (f)

Figure D.7: Results of applying the proposed method on *unr730* image. (a) Original image adopted from *http://ipagwww.med.yale.edu/∼staib*. (b) Single–layer representation. (c) Two–layer representation. (d) Three–layer representation. (e) Four–layer representation. (f) Five–layer representation.

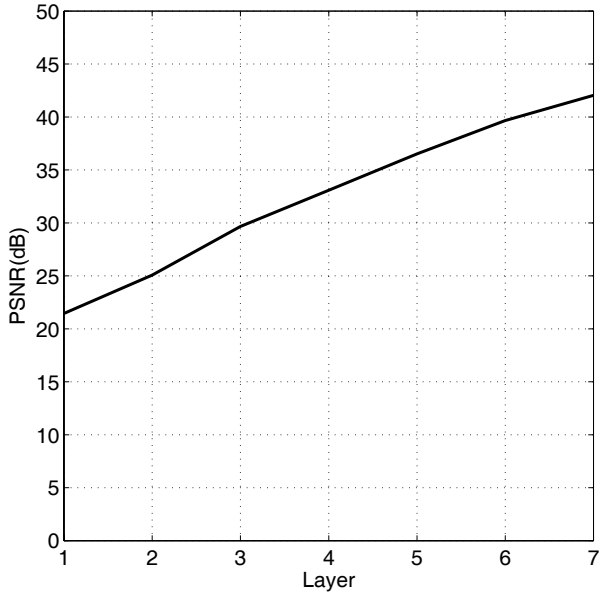


Figure D.8: PSNR versus layer number for *unr730* image.

approach results in a binary (single–layer) representation of an image, while the proposed method is able to give the representation of the given image in any number of layers. In the computational side, both the *Otsu*'s method and the proposed method rely on analytical search for a proper threshold. While the proposed method utilizes a logarithmic–order search algorithm, the *Otsu*'s approach searches an interval from the beginning bin–by–bin to find the optimum threshold.

D.4 Generalization

Assume that $x_1, cdots, x_n \in R$ are given. Also, assume that we need approximate x_is using a single layer. As such, the problem is to find the values $g_1, \cdots, g_n \in \{0, 1\}$

and the two real values r and s, for which $\tilde{x}_{1i} = rg_i + s(1 - g_i)$ is an appropriate approximation. Here, appropriateness is measured in terms of the following objective function which is to minimized,

$$\Delta = \sum_{i=1}^{n} [x_i - rg_i - s(1 - g_i)]^2. \tag{D.4.23}$$

Where, $f : R \to R^+ \cup \{0\}$ is a continuous differentiable function which is increasing in $R^+ \cup \{0\}$. First note that $\forall x \in R, f(x) = f(|x|)$. For an arbitrary value of i we have,

$$g_i = 1 \leftrightarrow f(|x_i - r|) \leq f(|x_i - s|), \tag{D.4.24}$$

or equivalently,

$$g_i = 1 \leftrightarrow |x_i - r| \leq |x_i - s|. \tag{D.4.25}$$

Assuming $s > r$, (D.4.25) turns into,

$$g_i = 1 \leftrightarrow x_i \leq \frac{1}{2}(r + s). \tag{D.4.26}$$

Now, lets find the values of r and s. We have,

$$\frac{\partial \Delta}{\partial r} = - \sum_{x_i \leq \frac{r+s}{2}} f'(x_i - r) = 0, \tag{D.4.27}$$

$$\frac{\partial \Delta}{\partial r} = - \sum_{x_i > \frac{r+s}{2}} f'(x_i - s) = 0. \tag{D.4.28}$$

For an arbitrary function $g : R \to R$ we define,

$$E_g\{x\} \equiv \arg_{r \in R} \sum_{x \in X} g(x - r) = 0. \tag{D.4.29}$$

For example, for $g(x) \equiv x$, E_g turns into the conventional average. Using this notation, (D.4.27) and (D.4.28) are converted to,

$$E_{f'}\{x_i \leq \theta\} + E_{f'}\{x_i > \theta\} = 2\theta, \tag{D.4.30}$$

solution of what gives the appropriate $\theta = \frac{r+s}{2}$.

Lets try to compute $E_g(X)$ for an arbitrary function g and a set X. Here, we focus on the continuous functions g satisfying $g(-x) = -g(x)$ and $g \geq 0 \rightarrow g(x) \geq 0$. Lets define,

$$G_X(r) = \sum_{x \in X} g(x - r) = \sum_{x \in X, x > r} g(x - r) - \sum_{x \in X, x \leq r} g(r - x). \tag{D.4.31}$$

Now,

$$G_X\Big(\min\{X\}\Big) > 0, G_X\Big(\max\{X\}\Big) < 0. \tag{D.4.32}$$

Hence, using bi–section, the value of r satisfying $G_X(r) = 0$ is approximated. Returning to the main problem, we were faced with an odd function so,

$$f'(-x) = \lim_{\Delta x \rightarrow 0} \frac{f(-x + \Delta x) - f(-x)}{\Delta x} = \\ - \lim_{\Delta x \rightarrow 0} \frac{f(x - \Delta x) - f(x)}{-\Delta x} = -f'(x). \tag{D.4.33}$$

As for $x \geq 0$, $f'(x)$ is positive, the method of bi–section will find the value of $E_{f'}\{X\}$ for any set $X \subset R$. For the set $X \subset [a, b]$ for finding $E_{f'}\{X\}$ with the precision of $2^{-\alpha}(b - a)$, α operations would be enough. Denoting the cost of computing f' by τ, the cost equals $(\alpha + 2)n\tau$.

Lets return to the main problem of solving (D.4.30). Thus, we are seeking for the zero point of the function,

$$F_X(\theta) = E_{f'}\{x_i \leq \theta\} + E_{f'}\{x_i > \theta\} - 2\theta. \tag{D.4.34}$$

Note that,

$$\lim_{\theta \to \min\{X\}^+} E_{f'}\{x \in X, x < \theta\} = \min\{X\}, \tag{D.4.35}$$

and

$$\lim_{\theta \to \max\{X\}^-} E_{f'}\{x \in X, x > \theta\} = \max\{X\}. \tag{D.4.36}$$

Hence,

$$F_X\Big(\min\{X\}\Big) = E_{f'}\{x \in X\} - \min\{X\}, \tag{D.4.37}$$

$$F_X\Big(\max\{X\}\Big) = E_{f'}\{x \in X\} - \max\{X\}, \tag{D.4.38}$$

We claim that for every function f with the conditions described above,

$$\min\{X\} \leq E_{f'}\{x \in X\} \leq \max\{X\}. \tag{D.4.39}$$

Note that if (D.4.39) is proved then $F_X\Big(\min\{X\}\Big) \geq 0$ and $F_X\Big(\max\{X\}\Big) \leq 0$ and so finding the zero of F_X would be possible using bi–section. To prove (D.4.39) note that,

$$r \leq \min\{X\} \to x_i - r \geq 0 \to f'(x_i - r) \geq 0 \to \sum_{x \in X} f'(x - r) \geq 0, \tag{D.4.40}$$

$$r \geq \max\{X\} \to r - x_i \geq 0 \to f'(r - x_i) \geq 0 \to \sum_{x \in X} f'(x - r) \leq 0. \tag{D.4.41}$$

Hence, the zero point of $\sum_{x \in X} f'(x-r)$ should be in the interval $[\min\{X\}, \max\{X\}]$, proving (D.4.39).

For a specific value of θ to compute (D.4.30) one should compute both $E_{f'}\{x_i \leq \theta\}$ and $E_{f'}\{x_i > \theta\}$. Assuming that $X \subset [a, b]$ and that $\theta = a\beta + b(1-\beta)$ $(0 \leq \beta \leq 1)$ finding the two above measures with the precision of $2^{-\alpha}$ multiplied by interval length elapses,

$$(\alpha+2)n\tau\beta + (\alpha+2)n\tau(1-\beta) = (\alpha+2)n\tau, \tag{D.4.42}$$

flops (assuming that the points are almost uniformly spread). For finding θ with the precision of $2^{-\alpha}(b-a)$ the total number of flops will get $(\alpha+2)^2 n\tau$.

Bibliography

[1] *Family safe media*, www.familysafemedia.com/pornography_ statistics.html.

[2] *Image after color image archive*, www.imaeafter.com.

[3] *Internet pornography: Are children at risk?*, www.nap.edu/ catalog/ 10261.html? onpi–webextra–050202.

[4] *Cie. international lighting vocabulary*, CIE Publications, 1989.

[5] *Itu–r recommendation bt–601–5: Studio encoding parameters of digital television for standard 4:3 and widescreen 16:9 aspect ratios*, www.itu.ch/, Geneva, 1994.

[6] *White house 'couples' set off indecency program*, Iowa City Press Citizen (1996).

[7] *Internet law and regulation*, FT Tax & Law (Graham J. H. Smith, ed.), London, UK, 1997, pp. 21–27.

[8] *Internet filter review*, www.internetfilterreview.com/ internet-pornography-statistics.html, 2003.

[9] *Websense^{TM}*, www.websense.com/ products/ resources/ wp/ hr_wp.pdf, 2003.

[10] *What is the future for watermarking (part i)*, IEEE Signal Processing Magazine (2003), 55–59.

[11] *What is the future for watermarking (part ii)*, IEEE Signal Processing Magazine (2003), 53–57.

[12] *Diopsis 740 dual–core dsp*, www.atmel.com, 2004.

[13] *Tms320c6713, tms320c6713b floating–point digital signal processors*, www.ti.com/, 2004.

[14] Arash Abadpour and Shohreh Kasaei, *Performance analysis of a pca–based color descriptor*, submitted to the CSI Journal of Computer Science and Engineering.

[15] ———, *Color image compression and watermarking using eigenimages*, Submitted to the Image & Vision Computing (2004).

[16] ———, *An efficient pca–based color transfer*, Visual Communication & Image Representation, under second review (2004).

[17] ———, *Fast color fpca–based clustering for image segmentation*, Submitted to Image & Vision Computing (2004).

[18] ———, *A new fast fuzzy color transfer method*, The 4th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2004) (Rome, Italy), 2004.

[19] ———, *A new fast fuzzy method for query region extraction from color images*, Submitted to the Pattern Recognition (2004).

[20] ———, *A new fpca–based fast segmentation method for color images*, The 4th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2004) (Rome, Italy), 2004.

[21] ———, *A new parametric linear adaptive color space and its pca–based implementation*, The 9th Annual CSI Computer Conference, CSICC (Tehran, Iran), 2004, pp. 125–132.

[22] ———, *A new pca–based compression method for natural color images*, IPM Workshop on Computer Vision (Tehran, Iran), 2004.

[23] ———, *A new pca–based robust color image watermarking method*, the 2nd IEEE Conference on Advancing Technology in the GCC: Challenges, and Solutions (Manama, Bahrain), 2004.

[24] ———, *A new principle component analysis based colorizing method*, Proceedings of the 12th Iranian Conference on Electrical Engineering (ICEE2004) (Mashhad, Iran), 2004.

[25] ———, *Performance analysis of three likelihood measures for color image processing*, IPM Workshop on Computer Vision (Tehran, Iran), 2004.

[26] ———, *Comprehensive evaluation of the pixel–based skin detection approach for pornography filtering in the internet resources*, Submitted to the International Symposium on Telecommunications (IST 2005) (Shiraz, Iran), 2005.

[27] ———, *Efficient color image and video copy protection*, Submitted to the 5th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2005) (Athens, Greece), 2005.

[28] ———, *Fast and robust color image watermarking using fpca clustering*, Submitted to the the IEE Proceedings Vision, Image & Signal Processing (2005).

[29] ———, *Fast registration of remotely sensed images*, 10th Annual CSI Computer Conference (CSICC2005) (Tehran, Iran), 2005.

[30] ———, *Multi–layer representation of gray–scale images*, Submitted to IEE Electronic Letters (2005).

[31] ———, *A new fast fuzzy method for query region extraction from color images*, 10th Annual CSI Computer Conference (CSICC2005) (Tehran, Iran), 2005.

[32] ———, *A new fast robust color image watermarking method using fpca clustering*, 10th Annual CSI Computer Conference (CSICC2005) (Tehran, Iran), 2005.

[33] ———, *A new tree decomposition method for color images*, 10th Annual CSI Computer Conference (CSICC2005) (Tehran, Iran), 2005.

[34] ———, *Novel color image compression method using eigenimages*, Third Conference on Machine Vision, Image Processing & Applications, (MVIP2005) (Tehran, Iran), 2005.

[35] ———, *Novel method for unsupervised fuzzy change detection in multispectral remotely sensed images*, Third Conference on Machine Vision, Image Processing & Applications, (MVIP2005) (Tehran, Iran), 2005.

[36] ———, *Novel method for unsupervised fuzzy change detection in multispectral remotely sensed images*, Submitted to the IAEEE Journal (2005).

[37] ———, *Pca–based skin detection for pornography filtering*, Submitted to the IEE Proceedings Communications (2005).

[38] ———, *Pixel–based skin detection for pornography filtering*, Tech. report, Sharif University of Technology, 2005.

[39] ———, *Principal color and its application to color image segmentation*, Submitted to the EURASIP Journal on Applied Signal Processing (2005).

[40] ———, *Realtime image–sequence recoloring*, Submitted to the 5th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2005) (Athens, Greece), 2005.

[41] ———, *Realtime image–sequence recoloring*, Submitted to the International Journal of Electronics and Communications (2005).

[42] ———, *Unsupervised, fast and efficient color image copy protection*, IEE Proceedings Communications, In Press (2005).

[43] ———, *An unsupervised fast color transfer method*, 13th Iranian Conference on Electrical Engineering (ICEE2005) (Zanjan, Iran), 2005.

[44] ———, *Wavelet–pca–based compression method for color images (invited paper)*, First Symposium on Wavelet and its Applications (Tehran, Iran), 2005.

[45] Arash Abadpour, Saeed Bagheri Shouraki, and Shohreh Kasaei, *A new method for agent–based color clustering*, World Congress on Fuzzy Logic, Soft Computing and Computational Intelligence Theories and Applications (IFSA2005) (Beijing, China), 2005.

[46] Y.I. Abdel-Aziz and H.M. Karara, *Direct linear transformation from comparator coordinates into object space coordinates in close range photogrammetry*, Proceedings of the ASP/UI Symposium on Close–Rage Photogrammetry (Urbana, Illinois), 1971, pp. 1–18.

[47] Yoshinoro Adachi, Akihiro Imai, Masahiro Ozaki, and Naohiro Ishii, *Extraction of face regions by using characteristics of color space and detection of face direction through an eigenspace*, Fourth IEEE International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies (Brighton, UK), 2000, pp. 393–396.

[48] J. Ahlberg, *A system for face localization and facial feature extraction*, Tech. Report LiTH–ISY–R–2172, Linkoping University, 1999.

[49] A. Albiol, L. Torres, and E.J. Delp, *Optimum color spaces for skin detection*, Proceedings of the International Conference on Image Processing, 2001, pp. 122–124.

[50] Alberto Albiol, Luis Torres, and Edward J. Delp, *An unsupervised color image segmentation algorithm for face detection applications*, Proceedings of the International Conference on Image Processing (Thessaloniki, Greece), 2001, pp. 681–684.

[51] Gilles Andre, Lucian Chiroiu, Catherine Mering, and Franck Chopin, *Building destruction and damage assessment after earthquake using high resolution optical sensors. the case of the gujarat earthquake of 26, 2001*, Unknown IEEE Conference., 2003.

[52] D. Androustsos, *Image retrieval using directional histogram*, Storage and Retrieval for image and video databases **1** (1998).

[53] D. Androutsos, K.N. Plataniotis, and A.N. Venetanopoulos, *Efficient indexing and retrieval of color image data using a vector–based approach*, Ph.D. thesis, University of Toronto, Toronto, ON, Canada, 1999.

[54] Will Archer Arentz and Bjorn Olstad, *Classifying offensive sites based on image content*, Computer Vision and Image Understanding **94** (2004), 295–310.

[55] Mauro Barni, Franco Bartolini, and Alessandro Piva, *Multichannel watermarking of color images*, IEEE Transaction on Circuits and Systems for Video Technology **12(3)** (2002), 142–156.

[56] B.Berlin and P. Kay, *Basic color terms, their universality and evolution*, University of California Press, Berkeley, 1969.

[57] A.M. Bensaid, *Partially supervised clustering for image segmentation*, Pattern Recognition **29(5)** (1996), 859871.

[58] K.B. Benson, *Television engineering handbook*, Mc Graw–Hill, New York, London, 1992.

[59] Kathleen M. Bergen, Daniel G. Brown, James R. Rutherford, and Eric J. Gustafson, *Development of a method for remote sensing of land–cover change 1980–200 in the ufs north central region using heterogeneous usgs luda and noaa avhrr 1km data*, Proceedings, International Geoscience and Remote Sensing Symposium (Toronto, CA), 2002.

[60] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*, Plenum Press, New York, 1981.

[61] J.C. Bezdek et al., *Fuzzy models and algorithms for pattern recognition and image processing*, Kluwer Academic Publishers, Boston, 1999.

[62] S. Birchfield, *Elliptical head tracking using intensity gradients and color histograms*, Proceedings of CVPR 98, 1998, pp. 232–237.

[63] Jeffrey A. Bloom, Ingemar J. Cox, TON Kalker, Jean-Paul M. G. Linnartz, Mathew L. Miller, and C. Brendan S. Traw, *Copy protection for dvd video*, Proceedings of the IEEE **87(7)** (1999), 1267–1276.

[64] Steven M. Boker, *The representation of color metrics and mappings in perceptual color space*, kiptron. psyc. virginia. edu/ steve_boker/ ColorVision2/ ColorVision2.html.

[65] M. Boninsegna and A. Bozzoli, *A tunable algorithm to update a reference image*, Signal Processing: Image Communication **16** (2000), 353–365.

[66] Robert J. Borns, *The internet: Privacy, the first amendment, and transnational communications; what's at stake*, Proceedings of FIE'96, 1996, pp. 1337–1341.

[67] Alison Bosson, Gavin C. Cawley, Yi Chan, and Richard Harvey, *Non–retrieval: Blocking pornographic images*, Proceedings of International Conference on the Challenge of Image and Video Retrieval, Lecture Notes in Computer Science (London), vol. 2383, Springer–Verlag, 2002, pp. 60–70.

[68] S. T. Bow, *Pattern recognition and image processing*, Marcel Dekker Inc., New York, 1992.

[69] Jason D. Brand and John S.D. Mason, *A comparative assessment of three approaches to pixel level human skin–detection*, Proceedings of the International Conference on Pattern Recognition, 2000, pp. 1056–1059.

[70] J.D. Brand and J.S.D. Mason, *A skin probability map and its use in face detection*, IEEE International Conference on Image Processing (ICIP01), 2001, pp. 1034–1037.

[71] Anne Wells Branscomb, *Internet babylon? does the carnegie mellon study of pornography on the information superhighway reveal a threat to the stability of society?*, Georgetown Law Journal (1995).

[72] David Brown, Ian Craw, and Julian Lewthwaite, *A som based approach to skin detection with application in real time systems*, Proceedings of the British Machine Vision Conference (Manchester, UK), 2001, pp. 491–500.

[73] L. Brown, *A survey of image registration techniques*, ACM Computing Surveys **24 No. 4** (1992), 325–376.

[74] James Bruce, Tucker Balch, and Manuela Veloso, *Fast and cheap color image segmentation for interactive robots*, Proceedings of IROS–2000 (Japan), 2000.

[75] Lorenzo Bruuzzone and Diego Fernandez Preito, *A bayesian approach to automatic change detection*, Proceedings of the IEEE 1999 Int. Geoscience and Remote Sensing Symposium (IGARSS99) (Hamburg, Germany), vol. 3, 1999, pp. 1816–1818.

[76] Lorenzo Bruzzone and Roberto Cossu, *Analysis of multitemporal remote–sensing images for change detection: Bayesian thresholding approaches*, Tech. Report DIT-02-0031, University of Trento, Department of Information and Communication Technology, 38050 Povo- Trento (Italy), 2003.

[77] Deborah Burke, *Cybersmut and the first amendment: A call for a new obscenity standard*, Harvard Journal of Law & Technology (1996).

[78] Christoph Busch, Wolfgang Funk, and Stephen Wolthusen, *Digital watermarking: From concepts to real–time video applications*, IEEE Computer Graphics and Applications (1999), 25–35.

[79] Darren Butler, Sridha Sridharan, and Vinod Chandran, *Chromatic colour spaces for skin detection using gmms*, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '02), 2002, pp. IV3620–IV3623.

[80] C. Cafforio and F. Rocca, *Methods for measuring small displacements of television images*, IEEE Transactions on Information Theory **22** (1976), 1973–1979.

[81] J. Cai, A. Goshtasby, and C. Yu, *Detecting human faces in color images*, International Workshop on Multi–Media Database Management Systems, 1998, pp. 124–131.

[82] M. J. Canty, I. Niemeyer, and Q.S. (Bob) Truong, *Change detection with multispectral satellite imagery: Application to nuclear verification*, Proceedings of the International Symposium on Spectral Sensing Research, ISSSR 2001 (CD-Rom) (Quebec City), 2001.

[83] Liang-Liang Cao, Xue-Long Li, Heng-Hai Yu, and Zheng-Kai Liu, *Naked people*

retrieval based on adaboost learning, Proceedings of the First International Conference on Machine Learning and Cybernetics (Beijing, China), 2002, pp. 1133–1138.

[84] Dragna Carevic and Terry Caelli, *Region–based coding of color images using karhunen–loeve transform*, Graphical Models and Image Processing **59(1)** (1997), 27–38.

[85] D. Chai and A. Bouzerdoum, *A bayesian approach to skin color classification in ycbcr color space*, Proceedings of IEEE Region Ten Conference (Kualu Lumpur, Malaysia), vol. 2, 2000, pp. 421–424.

[86] D. Chai and K. N. Ngan, *Locating facial region of a head–and–shoulders color image*, Proceedings of the Third International Conference on Automatic Face and Gesture Recognition, 1998, pp. 124–129.

[87] D. Chai and N. Ngan, *Face segmentation using skin–color map in video phone applications*, IEEE Transactions on Circuits and Systems for Video Technology **9(4)** (1999), 551–564.

[88] Douglas Chai, Lam Phung, and Abdesselam Bouzerdoum, *Skin color detection for face localization in human–machine communications*, International IEEE Symposium on Signal Processing and Its Application (ISSPA) (Kuala Lampur, Malaysia), 2001, pp. 343–346.

[89] Tamalika Chaira and A.K. Ray, *Fuzzy approach for color region extraction*, Pattern Recognition **24** (2003), 1943–1950.

[90] Y. Chan, R. Harvey, and J. Bagham, *Using colour features to block dubious images*, Proceedings of Eusipco 2000, 2000.

[91] Yi Chan, Richard Harvey, and Dan Smith, *Building systems to block pornography*, Challenge of Image Retrieval, BCS Electronic Workshops in Computing Series (CIR'99) (J. Eakins and D. Harper, eds.), Newcastle, 1999, pp. 34–40.

[92] Youngha Chang, Surguru Saito, and Masayuki Nakajima, *A framework for transfer colors based on the basic color categories*, Proceedings of the Computer Graphics International (CGI'03), IEEE, 2003.

[93] Chanchal Chatterjee and Vwani P. Roychowdhury, *Algorithms for coplanar camera calibration*, Machine Vision and Applications **12** (2000), 84–97.

[94] Q. Chen, H. Wu, and M. Yachida, *Face detection by fuzzy matching*, Proceedings of the Fifth IEEE International Conference on Computer Vision, 1995, pp. 591–596.

[95] Q. Chen, H. WU, and M. Yachida, *Face detection by fuzzy pattern matching*, Proceedings of the Fifth International Conference on Computer Vision, 1995, pp. 591–597.

[96] Qian Chen, Haiyuan Wu, , and Masahiko Yachida, *Face detection by fuzzy pattern matching*, IEEE 5th International Conference on Computer Vision, 1995, pp. 591–596.

[97] Tie Qi Chen and Yi Lu, *Color image segmentation – an innovative approach*, Pattern Recognition **35** (2002), 395–405.

[98] Tongbo Chen, Yan Wang, Volker Schilings, and Christoph Meinel, *Grayscale image matting and colorization*, Proceedings of ACCV2004 (Jeju Island, Korea), 2004, pp. 1164–1169.

[99] Shyi-Chyi Cheng and Shih-Chang Hsia, *Fast algorithm's for color image processing by principal component analysis*, Journal of Visual Communication and Image Representation **14** (2003), 184–203.

[100] T.W. Cheng, D.B. Goldgof, and L.O. Hall, *Fast fuzzy clustering*, Fuzzy Sets and Systems **1998** (1993), 4956.

[101] Chun-Hsien Chou and Tung-Lin Wu, *Embedding color watermarks in color images*, EURASIP Journal on Applied Signal Processing **1** (2001), 327–332.

[102] T.S. Chua, S.K. Lim, and H.K. Pung, *Content based retrieval of segmented images*, Proceedings of ACM International Conference on Multimedia, 1994.

[103] Tae-Yun Chung, Min-Suk Hong, Young-Num Oh, Dong-Ho Shin, and Sang-Hui Park, *Digital watermarking for copyright protection of mpeg2 compressed video*, IEEE Transactions on Consumer Electronics **44(3)** (1998), 895–901.

[104] Clifford Clausen and Harry Wechsler, *Color image compression using pca and backpropagation learning*, Pattern Recognition **33** (2000), 1555–1560.

[105] M. Collobert, R. Feraud, G. Le Tourneur, D. Bernier, J. E. Vaiallet, Y. Mahieux, and D.Collobert, *Listen: A system for locating and tracking individual speakers*, Proceedings of the International Conference on Automatic Face and Gesture Recognition, 1996, pp. 283–288.

[106] Jonathan Coopersmith, *Pornography, videotape, and the internet*, IEEE Technology and Society Magazine **Spring** (2000), 27–34.

[107] Symon Cotton, *Colour, colour spaces and the human visual system*, www.cs.bham.ac.uk/šdc, 1995.

[108] Scott A. Craver, Min Wu, and Bede Liu, *What can we reasonably expect from watermarks?*, 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (New Paltz, New York), 2001, pp. 223–226.

[109] J.L. Crowley and J.M. Bedrune, *Integration and control of reactive visual processes*, Proceedings of the Third European Conference on Computer Vision, vol. 2, 1994, pp. 47–58.

[110] J.L. Crowley and F. Berard, *Multi–modal tracking of faces for video communications*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 640–645.

[111] Xiaolong Dai and Siamak Khorram, *Development of a new automated land cover change detection system from remotely sensed imagery based on artificial neural networks*, Proceedings of 1997 IEEE International Geoscience and Remote Sensing Symposiums (IGARSS'97) (Singapore), 1997, pp. 1029–1031.

[112] Xiaoyan Dai and Iunji Maeda, *Fuzzy–based segmentation of color natural images*, IEEE ICSP'02, 2002, pp. 969–972.

[113] Y. Dai and Y. Nakano, *Extraction for facial images from complex background using color information and sgld matrices*, Proceedings of the First International Workshop on Automatic Face and Gesture Recognition, 1995, pp. 238–242.

[114] ———, *Face–texture model based on sgld and its application in face detection in a color scene*, Pattern Recognition **29(6)** (1996), 1007–1017.

[115] R.N. Dave and K. Bhaswan, *Adaptive fuzzy c–shells clustering and detection of ellipses*, IEEE Trans. Neural Networks **3(5)** (1992), 643662.

[116] M.G. Day and E.F. Gehringer, *Educators and pornography: the "unacceptable use" of school computers*, International Symposium on Technology and Society (ISTAS'02) (340–344), 2002.

[117] N. Diehl, *Object–oriented motion estimation and segmentation in image sequences*, IEEE Transactions on Image Processing **3** (1990), 1901–1904.

[118] A. Dimai and M. Striker, *Color indexing with weak spatial constraints*, Symposium in electronic imaging: Science and Technology-Storage and retrieval for image and video databases IV **2670** (1996), 29–41.

[119] Chen Ding, Chi-Hung Chi, Jing Deng, and Chun-Lei Dong, *Centralized content–based web filtering and blocking: How far can it go*, Proceeding of IEEE International Conference on Systems, Man and Cybernetics (SMC), 1999.

[120] Gwenael Doerr and Jean-Luc Dugelay, *A guide tour of video watermarking*, Signal Processing: Image Coomunication **18** (2003), 263–282.

[121] W. Doyle, *Operations useful for similarity–invariant pattern recognition*, Journal of Association for Computing Machinery **9(2)** (1962), 259–267.

[122] A. F. Drimbarean, P. M. Corcoran, M. Cuic, and V. Buzuloiu, *Image processing techniques to detect and filter objectionable images based on skin tone and shape recognition*, Proceedings of the International Conference on Consumer Electronics (ICCE), 2001, pp. 278–279.

[123] Yingzi Du, Chein-I Chang, and Paul David Thouin, *Unsupervised approach to color video thresholding*, Optical Engineering **43 (2)** (2004), 282–289.

[124] R.O. Duda and P.E. Hart, *Pattern classification and scene analysis*, Wiley, New York, 1973.

[125] J.C. Dunn, *A fuzzy relative of the isodata process and its use in detecting compact well–separated clusters*, Journal of Cybernetics **3** (1973), 32–57.

[126] Phillip Elmer-DeWitt, *Cyberporn*, Time magazine **146(1)** (1995), 1–10.

[127] F. Ennesser and G. Medioni, *Finding waldo, or focus of attention using local color information.*, IEEE Trans. on Pattern Analysis and Machine Intelligence (1995), 805–809.

[128] Christos Faloutsos, H.V. Jagadish, and Yannis Manolopoulos, *Analysis of the n–dimensional quadtree decomposition for arbitrary hyperrectangles*, IEEE Transaction on Knowledge and Data Engineering **9, No. 3** (1997), 373–383.

[129] Ayesha Farrukh, Arif Ahmad, M. Ijaz Khan, and Nadeem Khan, *Automated segmentation of skin–tone regions in video sequences*, Proceedings of the IEEE Students Conference (ISCON'02), 2002, pp. 122–128.

[130] Jeffrey E. Faucette, *The freedom of speech and a frightened university's censorship of sex on the internet*, Duke Law Journal (1995).

[131] Rogerio Schmidt Feris, Toefilo Emidio de Campos, and Roberto Marcondes Cesar Junior, *Detection and tracking of facial features in video sequences*, Lecture Notes in Artificial Intelligence **1793** (2002), 197–206.

[132] Stefan Feyrer and Andreas Zell, *Detection, tracking, and pursuit of humans with an autonomous mobile robot*, International Conference on Intelligent Robots and Systems 1999 (IROS 99) (Kyongju, Korea), 1999, pp. 864–869.

[133] Graham Finlayson and Gerald Schaefer, *Hue that is invariant to brightness and gamma*, Proceedings of British Machine Vision Conference (Manchester, UK).

[134] Chris S. Fischer and Lisa M. Levinen, *Monitoring california's hardwood rangelands using remotely sensed data*, Proceedings of the Fifth Oak Symposium (Oak Woodlands), 2001.

[135] K. Fishkin and B. Narsky, *A family of new algorithms for soft filling*, Computer Graphics **18(3)** (1984), 235–244.

[136] Margaret Fleck, David Forsyth, and Chris Bregler, *Finding naked people*, European Conference on Computer Vision (B. Buxton and R. Cipolla, eds.), Springer–Verlag, Berlin, Germany, 1996, pp. 593–602.

[137] J. Foley, A. Van Dam, S. Feiner, and J. Hughes, *Computer graphics: Principles and practice*, Addison–Wesley, Reading MA, 1990.

[138] J.D. Foley and A. Van Dam, *Fundamentals of interactive computer graphics, the system programming series,*, Addison–Wesley, Reading, MA., 1982, Reprinted 1984 with corrections.

[139] Adrian Ford and Alan Roberts, *Colour space conversions*, ajoec1@wmin.ac.uk, 1998.

[140] David A. Forsyth and Margaret Fleck, *Automatic detection of human nudes*, International Journal of Computer Vision **32(1)** (1999), 63–77.

[141] David A. Forsyth and Margaret M. Fleck, *Identifying nude pictures*, Third IEEE Workshop on the Applications of Computer Vision (WACV '96) (Sarasoto, Florida), 1996, pp. 103–108.

[142] H. Frigui, *Interactive image retrieval using fuzzy sets*, Pattern Recognition Letters **22** (2001), 1021–1031.

[143] Hsin-Chia Fu, P.S. Lai, R.S. lou, and H.-T. Pao, *Face detection and eye localization by neural network based color segmentation*, Proceedings of the 2000 IEEE Signal Processing Society Workshop, 2000, pp. 507–516.

[144] K.S. Fu and J.K. Mui, *A survey on image segmentation*, Pattern Recognition **13** (1981), 3–16.

[145] Shinji Fukuma, Masahiro Iwahashi, and Noriyoshi Kambayashi, *Lossless color coordinate transform for lossless color image coding*, 1998, pp. 595–598.

[146] B. Funt and G. Finlayson, *Color constant color indexing*, IEEE Trans. On Pattern Analysis and Image Processing **17** (1995), 522–529.

[147] C. Garcia and G. Tziritas, *Face detection using quantized skin color regions merging and wavelet packet analysis*, IEEE Transactions on Multimedia **1(3)** (1999), 264–277.

[148] I. Gath and A.B. Geva, *Unsupervised optimal fuzzy clustering*, IEEE Transaction on Pattern Analysis Machine Intelligence **11(7)** (1989), 773–781.

[149] David Gering, *Linear and nonlinear data dimensionality reduction*, MIT Area Exam, www.ai.mit.edu/people/ gering/areaexam/, 2002.

[150] S.A.M. Gilani, I. Kostopoulos, and A.N. Skodras, *Color image–adaptive watermarking*, 14th Int. Conf. on Digital Signal Processing(DSP2002) (Santorini, Greece), vol. 2, 2002, pp. 721–724.

[151] Philip E. Gill, Walter Murray, and Margaret H. Wright, *Numerical linear algebra and optimization*, Addison–Wesley Publishing Company, Redwood City, California, 1991.

[152] Giovani Gomez, *On selecting colour components for skin detection*, Proceedings of the ICPR, 2000, pp. 961–964.

[153] R.C. Gonzalez and P. Wintz, *Digital image processing*, Addison–Wesley Publications, Reading, MA., 1987.

[154] A. Goshtasby and G.C. Stockman, *Point pattern matching using convex hull edges*, IEEE Transactions on Systems, Man and Cybernetics **15** (1985), 631–637.

[155] H. P. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan, *Multi–modal system for locating heads and faces*, Proceedings of the Second International Conference on Automatic Face and Gesture Recognition (Killington, Vermont), 1996, pp. 88–93.

[156] Gary R. Greenfield and Donald H. House, *Image recoloring induced by palette color associations*, Journal of WSCG'03 **11(1)** (2003), 3–7, 2003.

[157] P. Greenfield, P. Rickwood, and H. Tran, *Effectiveness of internet filtering software products*, CSIRO Mathematical and Information Science, 2001.

[158] I. Grinias and G. Tziritas, *A semi–automatic seeded region growing algorithm for video object localization and tracking*, Signal Processing: Image Communication **16** (2001), 977–986.

[159] Britt Grisworld, *Stopping webway rubbery: On–line image protection in the digital age*, bgriswol@pop200.gsfc.nasa.goc, Visited 2004.

[160] Wendy M. Grossman, *net.wars*, NYU Press, 1997.

[161] D. E. Gustafson and W. C. Kessel, *Fuzzy clustering with a fuzzy covariance matrix*, Proceedings of the IEEE CDC (San Diego, CA), vol. 2, 1979, pp. 761–766.

[162] Palus H., *Representation of color images in different color spaces*, The color image processing handbook (S.J. Sangwine and R.E.N. Horne, eds.), Chapman and hall, London, 1998, pp. 67–90.

[163] M. Hammami, Y. Chahir, and L. Chen, *Webguard: Web based adult content detection and filtering system*, Proceedings of the International Conference on Web Intelligence (WI'2003), 2003, pp. 574 – 578.

[164] M. Hammami, D. Tsishkou, and Chen Liming, *Adult content web filtering and face detection using data-mining based skin-color model*, Proceedings of the International Conference on Multimedia and Expo (ICME'04), 2004, pp. 403–406.

[165] Allan Hanbury, *The taming of the hue, saturation and brightness colour space*, 7th Computer Vision Winter Workshop (Bad Aussee, Austria).

[166] Pengwei Hao and Qingyun Shi, *Comparative study of color transforms for image coding and derivation of integer reversible color transform*, 2000, pp. 224–227.

[167] R.M. Haralick and L.G. Shapiro, *Image segmentation techniques*, Computer Vision, Graphics, and Image Processing **29 (1)** (1985), 100–132.

[168] H. Hellendorn and D. Driankov, *Fuzzy: Model identification*, Springer Vrlag, Heidelberg, 1998.

[169] Cormac Herley, *Why watermarking is nonsense*, IEEE Signal Processing Magazine (2002), 10–11.

[170] N. Herodotou, K.N. Plataniotis, and A.N. Venetsanopoulos, *Automatic location and tracking of the facial region in color video sequences*, Signal Processing: Image Communication **14** (1999), 359–388.

[171] W.H. Ho and P.A. Watters, *Statistical and structural approaches to filtering internet pornography*, IEEE International Conference on Systems, Man and Cybernetics, 2004, pp. 4792–4798.

[172] M. Holm, *Towards automatic rectification of satellite images using feature based matching*, Proceedings of the International Geoscience and Remote Sensing Symposium IGARSS91 (Espoo, Finland), 1991, pp. 2439–2442.

[173] Katsuhiro Honda, Nobukazu Sugiura, and Hidetomo Ichihashi, *Robust local principal component analyzer with fuzzy clustering*, IJCNN 2003 Conference Proceedings, 2003, pp. 732–737.

[174] Takahiko Horiuchi, *Colorization algorithms using probabilistic relaxation*, Image and Vision Computing **22** (2004), 197–202.

[175] Takahiko Horiuchi and Sayaka Hirano, *Colorization algorithm for grayscale image by propagating seed pixels*, IEEE International Conference on Image Processing (ICIP'03), 2003, pp. 457–460.

[176] H. Hotteling, *Analysis of a complex of statistical variables into principal components.*, Journal of Educational Psychology **24** (1933), 417–441.

[177] Ming-Shing Hsieh, Din-Chang Tseng, and Yong-Huai Huang, *Hiding digital watermarks using multiresolution wavelet transform*, IEEE Transaction on Industrial Electronics **48(5)** (2001), 875–882.

[178] Y.C. Hsieh, D.M. McKeown, and F.P. Perlant, *Performance evaluation of scene registration and stereo matching for cartographic feature extraction*, IEEE Transactions on Pattern Analysis and Machine Intelligence **14** (1992), 214–237.

[179] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K. Jain, *Face detection in color images*, IEEE Trans. Pattern Analysis and Machine Intelligence **24(5)** (2002), 696–702.

[180] W. Hsu, T.J. Chua, and H.K. Pung, *An integrated color spatial approach to content–based retrieval*, Proceedings of ACM International Conference on Multimedia, 1995.

[181] Martin Hunke and Alex Waibel, *Face locating and tracking for human-computer interaction*, IEEE Computer, 1994, pp. 1277–1281.

[182] Aapo Hyvarinen, Juha Karhunen, and Erkki Oja, *Independent component analysis*, John Wiley & Sons, Inc., 2001.

[183] Aapo Hyvrinen and E. Oja, *Independent component analysis: Algorithms and applications*, Neural Networks **13(4–5)** (2000), 411–430.

[184] S. Ioffe and D. A. Forsyth, *Probabilistic methods for finding people*, International Journal of Computer Vision **43(1)** (2001), 45–68.

[185] Sergey Ioffe and David Forsyth, *Finding people by sampling*, IEEE International Conference on Computer Vision (Corfu, Greece), 1999.

[186] N.S. Iyer and A. Kandel, *Feature–based fuzzy classification for interpretation of mammograms*, Fuzzy Sets and Systems **114** (2000), 271280.

[187] A.K. Jain and R.C. Dubes, *Algorithms for clustering*, Prentice–Hall, NJ, 1998.

[188] A.K. Jain, M.N. Murty, and P.J. Flynn, *Data clustering: a review*, ACM Computer Surveys **31(3)** (1999), 264323.

[189] Anil K. Jain, *Fundamentals of digital image processing*, Prentice–Hall International Inc., Republic of Sigapore, 1989.

[190] R. Jain, R. Kasturi, and B.G. Schunck, *Machine vision*, McGraw–Hill Inc., New York NY, 1995.

[191] Sriram Jayaram, Stephen Schmugge, Min C. Shin, and Lenoid V. Tsap, *Effect of color space transformation, the illuminance component, and color modelling on skin detection*, 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04), 2004.

[192] T.S. Jebara and A. Pentland, *Parameterized structure from motion for 3d adaptive feedback tracking of faces*, Proc. IEEE Conference Computer Vision and Pattern Recognition, 1997, pp. 144–150.

[193] T.S. Jebara, K. Russell, and A. Pentland, *Mixtures of eigenfeatures for real-time structure from texture*, Proc. Sixth IEEE International Conference on Computer Vision, 1998, pp. 128–135.

[194] Chi-Yoon Jeong, Jong-Sung Kim, and Ki-Sang Hong, *Appearance–based nude image detection*, Proceedings of 17–th International Conference on Pattern Recognition (ICPR'04) (Cambridge, UK), 2004.

[195] Feng Jiao, Wen Gao, Lijuan Duan, and Guoqin Cui, *Detecting adult image using multiple features*, Proceedings of International Conferences on Info-tech and Info-net ICII'01 (Beijing, China), 2001, pp. 378–383.

[196] Tieniu Tan Weiming Hu Jinfeng Yang, Zhouyu Fu, *A novel approach to detecting adult images*, Proceedings of 17–th International Conference on Pattern Recognition (ICPR'04) (Cambridge, UK), 2004.

[197] I. T. Jolliffe, *Principal component analysis*, Springer–Verlag, Heidelberg–New York, 1986.

[198] Michael J. Jones and James M. Rehg, *Statistical color models with application to skin detection*, Proceedings of Computer Vision and Pattern Recognition (CVPR'99), 1999, pp. 274–280.

[199] ———, *Detecting adult images*, HP Cambridge Research Laboratory, crl.research.compaq.com/ projects/ vision/ adult_detection.html, 2002.

[200] ———, *Skin color modelling and detection*, HP Cambridge Research Laboratory, crl.research.compaq.com/ projects/ vision/, 2002.

[201] L. Jordao, M. Perrone, J. Costeira, and J. Santos-Victor, *Active face and feature tracking*, Proceedings of the 10th International Conference on Image Analysis and Processing, 1999, pp. 572–577.

[202] M.S. Kamel and S. Selim, *New algorithms for solving the fuzzy clustering problem*, Pattern Recognition **27(3)** (1994), 421428.

[203] Xiangui Kang, Jiwu Huang, Yun Q. Shi, and Yan Lin, *A dwt–dft composite watermarking scheme robust to both affine transform and jpeg compression*, IEEE Transactions on Circuits and Systems for Video Technology **13(8)** (2003), 776–786.

[204] K. Karmann, A Barandt, and R. Gerl, *Moving object segmentation based on adaptive reference images*, European Signal Processing Conference, 1990.

[205] S. Kasaei, M. Dericher, and B. Biashash, *A novel fingerprint image compression technique using wavelet packets and pyramid lattice vector quantization*, IEEE Transactions on Image Processing **11(12)** (2002), 1365–1378.

[206] M. Kass, A. Witkin, and D. Trezopoulos, *Snake: Active contour models*, International Journal of Computer Vision **1(4)** (1987), 321–331.

[207] S. Katzenbeisser and A.P. Petitcolas, *Information hiding techniques for steganography and digital watermarking*, Artech House Inc., 2000.

[208] Shinjiro Kawato and Jun Ohya, *Automatic skin–color distribution for face detection and tracking*, Proceedings of ICSP2000, 2000, pp. 1415–1418.

[209] M. Kendall, *Multivariate analysis*, Charlas Griffin and Co., 1975.

[210] Changick Kim, *Content–based image copy detection*, Signal Processing: Image Communication **18** (2003), 169–184.

[211] Changick Kim and Jenq-Neng Hwang, *Fast and automatic video object segmentation and tracking for content–based applications*, IEEE Transaction on Circuits and Systems for Video Technology **12 (2)** (2002), 122–129.

[212] Dae-Won Kim, Kwang H. Lee, and Doheon Lee, *A novel initialization scheme for the fuzzy c–means algorithm for color clustering*, Pattern Recognition Letters **25** (2004), 227237.

[213] J. Kim and T. Chen, *combining static and dynamic feature using neural networks and edge diffusion fir video object extraction*, IEE Proceedings in Visual Image Signal Processing **150(3)** (2003), 160–167.

[214] Jinsang Kim and Tom Chen, *Multiple feature clustering for image sequence segmentation*, Pattern Recognition Letters **22** (2001), 1207–1217.

[215] Jong Bae Kim and Hang Joon Kim, *Efficient region–based motion segmentation for a video monitoring system*, Pattern Recognition Letters **24** (2003), 113–128.

[216] Sang-Hoon Kim and Hyoung-Gon Kim, *Face detection using multi–modal information*, Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000 (Europole Atrium, Grenoble, France), 2000, p. 14.

[217] Sanh-Hoon Kim, Nam-Kyu Kim, Sang Chul Ahn, and Hyoung-Gon Kim, *Object oriented face detection using range and color information*, Proceedings of the Third International Conference on Automatic Face and Gesture Recognition, 1998, pp. 76–81.

[218] Rich Kjeldsen and John Kender, *Finding skin in color images*, Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996, pp. 312–317.

[219] Gurdun J. Klinker, Steven A. Shafer, and Takeo Kanade, *The measurement of highlights in color images*, International Journal of Computer Vision **2** (1988), 7–32.

[220] ———, *A physical approach to color image understanding*, International Journal of Computer Vision **4** (1990), 7–38.

[221] R. Krishnapuram, H. Frigui, and O. Nasraoui, *Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximationpart i*, IEEE Trans. Fuzzy Systems **3(1)** (1995), 4460.

[222] ———, *Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximationpart ii*, IEEE Trans. Fuzzy Systems **3(1)** (1995), 4460.

[223] R. Krishnapuram and J.K. Keller, *A possibilistic approach to clustering*, IEEE Transaction on Fuzzy Systems **1** (1993), 98–110.

[224] R. Krishnapuram and J.N. Keller, *The possibilistic c–means algorithms: Insights and recommendations*, IEEE Transactions on Fuzzy Systems **4** (1996), 385–393.

[225] R. Krishnapuram and J. Kim, *Clustering algorithms based on volume criteria*, IEEE Trans. Fuzzy Systems **8(2)** (1995), 228236.

[226] R. Krishnapuram, O. Nasraoui, and H. Frigui, *The fuzzy c–spherical shells algorithm: a new approach*, IEEE Trans. Neural Networks **3(5)** (1992), 663671.

[227] Tony Krone, *A typology of online child pornography offending*, Australian Government, Australian Institute of Criminology, Trends and Issues in Crime and Criminal Justice, No. 279, *www.ahtcc.gov.au*, 2004.

[228] Deepa Kundur and Kannan Marthik, *Video fingerprinting and encryption principles for digital rights management*, Proceedings of the IEEE **92 (6)** (2004), 918–932.

[229] M. Kutter, F. Jordan, and F. Bossen, *Digital watermarking of color images using amplitude modulation*, Electronic Imaging **7(2)** (1998), 326–332.

[230] M Kutter and F. A. P. Petitcolas, *A fair benchmark for image watermarking systems*, Electronic Imaging'99, Security and Watermarking of Multimedia Content (San Jose, Ca, USA), 1990.

[231] Martin Kutter and Stefan Winkler, *A vision–based masking model for spread–spectrum image watermarking*, IEEE Transaction on Image Processing **11(1)** (2002), 16–25.

[232] Gerhard C. Langelaar, Iwan Setyawan, and Reginald L. Lagendijk, *Watermarking digital image and video data, a state–of–the–art overview*, IEEE Signal Processing Magazine (2000), 20–46.

[233] S. Ledley, M. Nuas, and T. Golab, *Fundamentals of true color image processing*, tenth International Conference on Pattern Recognition (Atlantic City), 1990, pp. 791–795.

[234] Ho Young Lee, Ho Heun Lee, and Yeong Ho Ha, *Spatial color descriptor for image retrieval and video segmentation*, IEEE Transaction on Multimedia **5(3)** (2003), 358–367.

[235] M. Leeser, N. Kitaryeva, and J. Crisman, *Spatial and color clustering on an fpga–based computer system*, Proc. SPIE, 1998, p. 2533.

[236] Jacek M. Leski, *Fuzzy c–varieties/elliptotypes clustering in reproducing kernel hilbert space*, Fuzzy Sets and Systems **141** (2004), 259–280.

[237] Shu-Hung Leung, Shi-Lin Wang, and Wing-Hong Lau, *Lip image segmentation using fuzzy clustering incorporating an elliptic shape function*, IEEE Transaction on Image Processing **13(1)** (2004), 51–62.

[238] Anat Levin, Dani Lischinski, and Yair Weiss, *Colorization using optimization*, ACM Transactions on Graphics (2004).

[239] H. Levkowitz and G.T. Herman, *Color scales for image data*, IEEE Computer Graphics and Application (1992).

[240] Hui Li, B. S. Manjunath, and Sanjit K. Mitra, *A contour–based approach to multisensor image registration*, IEEE Transactions on Image Processing **4 No. 3** (1995), 320–333.

[241] Lituan Li, Weimin Huang, Irene Y. H. Gu, and Qi Tian, *Foreground object detection in changing background based on color co–occurrence statistics*, Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV'02).

[242] S.Z. Li and J. Kittlerand M. Petrou, *Matching and recognition of road networks from aerial images*, Proceedings of the Second European Conference on Computer Vision ECCV92 (St Margherita, Italy), 1992, pp. 857–861.

[243] J. O. Limb, C.B. Rubinstein, and J.E. Thompson, *Digital coding of color video signals– a review*, IEEE Transaction of Communication **25(11)** (1977), 1349–1385.

[244] Yu-Chun Lin, Hung-Wei Tseng, and Chiou-Shann Fuh, *Pornography detection using support vector machine*, 16th IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP 2003) (Kinmen, ROC), 2003, pp. 123–130.

[245] Fuhui Long, Dagan Feng, and Hanchuan Peng, *Extracting semantic video objects*, IEEE Computer Graphics and Applications **January/Feburary** (2001), 48–55.

[246] L. Lucchese and S.K. Mitra, *Color image segmentation: A state–of–the–art survey (invited paper)*, Image Processing, Vision, and Pattern Recognition, Proc. of the Indian National Science Academy (INSA–A) **67, A, No. 2** (2001), pp. 207–221.

[247] Kai-Kuang Ma and Junxian Wang, *Color distance histogram: A novel descriptor for color image segmentation*, Seventh International Conference on Control, Automation, Robotics, and Vision (ICARCV'02) (Singapore), 2002, pp. 1228–1232.

[248] Lindsay W. MacDonald and M. Ronnier Luo, *Color image science, exploiting digital media*, John Wiley and Sons Ltd, 2002.

[249] Maurice Maes, Tom Kalker, Jean-Paul M.G. Linnartz, Joop Talstra, Geert F. G. Depovere, and Jaap Haitsma, *Digital watermarking for dvd copy protection,*

what issues play a role in designing and effective system, IEEE Signal Processing Magazine (2000), 47–57.

[250] H. Maitre and Y. Wu, *Improving dynamic programming to solve image registration*, Pattern Recognition **20** (1987), 443–462.

[251] B.S. Manjunath, Jens-Rainer Ohm, Vinod Vasuderan, and Akio Yamada, *Color and texture descriptors*, IEEE Transaction on Circuits and Systems for Video Processing **11 No. 6** (2001), 703–715.

[252] F. Marques and V. Vilplana, *A morphological approach for segmentation and tracking of human faces*, In International Conference on Pattern Recognition (ICPR00), 2000, pp. 5064–5068.

[253] A. M. Massone, F. Masulli, and A. Petrosini, *Fuzzy clustering algorithms and landsat images for detection of waste areas: A comparison*, Advances in Fuzzy Systems and Intelligent Technologies (2000), 165–175.

[254] F. Masulli, A.M. Massone, and A. Schenone, *Fuzzy clustering methods for the segmentation of multimodal medical volumes (invited)*, Fuzzy Systems in Medicine. Series Studies in Fuzziness and Soft Computing, editor J. Kacprzyk (P.S. Szczepaniak, P.J.G. Lisboa, and S. Tsumoto, eds.), Springer–Verlag, Heidelberg (Germany), 2000, pp. 335–350.

[255] Nebu John Mathai, Deepa Kundur, , and Ali Sheikholeslami, *Hardware implementation perspectives of digital video watermarking algorithms*, IEEE Transactions on Signal Processing **51(4)** (2003), 625–638.

[256] Masashi Matsuoka and Fumio Yamazaki, *Application of the damage detection method using sar intensity images to recent earthquakes*, Proceedings of the International Geoscience and Remote Sensing Symposium, IEEE,, 2002, pp. CD–ROM.

[257] S. McKenna, S. Gong, and Y. Raja, *Modelling facial colour and identity with gaussian mixtures*, Pattern Recognition **31** (1998), 1883–1892.

[258] Michael D. Mehta and Dwaine Plaza, *Content analysis of pornographic images available on the internet*, The Information Society **13(2)** (1997), 153–161.

[259] B. Menser and M. Wien, *Segmentation and tracking of facial regions in color image sequences*, Proceedings of SPIE Visual Communications and Image Processing, 2000, pp. 731–740.

[260] Hajime Mitomi, Masahi Matsuoka, Fumio Yamazaki, Hitoshi Taniguchi, and Yujiro Ogawa, *Determination of the areas with building damage due to the 1995 kobe earthquake using airborne mss images*, IEEE International Geoscience and Remote Sensing Symposium, CD–ROM, 2002.

[261] T. Mitsunaga, T. Yokoyama, and T. Totsuka, *Autokey: Human assisted key extraction*, SIGGRAPH95, 1995, pp. 265–272.

[262] Y. Miyake, H. Saitoh, H. Yaguchi, and N. Tsukada, *Facial pattern detection and color correction from television picture for newspaper printing*, Imaging Technology **16(5)** (1990), 165–169.

[263] Pierre Moulin, *Comments on "why watermarking is nonsense"*, IEEE Signal Processing Magazine (2003), 57–59.

[264] Thomas A. Mysliwiec, *Fingermouse: A freehand computer pointing interface*, Tech. Report VISLab-94-01, University of Illinois at Chicago.

[265] Masanori Nakamura, Mitsuteru Sakamoto, Shigeru Kakumoto, and Yukio Kosugi, *Stabilizing the accuracy of change detection from geographic images by multi–levelled exploration and selective smoothing*, Proceedings of GIS2003 (Vancouver), 2003.

[266] Huzefa Neemuchwala, Alfred Hero, , and Paul Carson, *Image registration using alpha–entropy measures and entropic graphs*, European Journal on Signal Processing (www.elsevier.nl/locate/sigpro), Special Issue on: Content–based Visual Information Retrieval. (2004).

[267] Athanasios Nikoladis and Ioannis Pitas, *Robust watermarking of facial images based on salient geometric pattern matching*, IEEE Transaction on Multimedia **2(3)** (2000), 172–184.

[268] Dmitry O. Nikolaev and Petr O. Nikolayev, *Linear color segmentation and its implementation*, Computer Vision and Image Understanding **94** (2004), 115–139.

[269] I. Noda, S. Suzuki, H. Matsubara, M. Asada, and H. Kitano., *Robocup–97: The first robot world cup soccer games and conferences.*, AI Magazine (1998).

[270] N. Nojic and K. K. Pang, *Adaptive skin segmentation for head and shoulder video sequence*, SPIE VCIP'2000 (Perth, Australia), 2000.

[271] J.-M Odobez and P. Bouthemy, *Robust multiresolution estimation of parametric motion models*, Visual Communication and Image Representation **6** (1995), 348–365.

[272] V. Ogle and M. Stonebaker, *Chabot: Retrieval form a relational database of images*, IEEE Computer **28(9)** (1995), 40–48.

[273] Y. I. Ohta, Y. Kanade, and T. Saki, *Color information for region segmentation*, Computer Graphics and Image Processing **13, No.3** (1980), 222–241.

[274] N. Oliver, A. Pentland, and F. Berard, *Lafer: Lips and face real time tracker*, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1997, pp. 123–129.

[275] N. Otsu, *A threshold selection method from gray level histograms*, IEEE Transactions on Systems, Man, and Cybernetics **SMC–9** (1979), 62–66.

[276] N.R. Pal and S.K. Pal, *A review on image segmentation techniques*, Pattern Recognition **26(9)** (1993), 12771294.

[277] Henryk Palus, *Color spaces*, Chapmann and Hall, 1st edition, London, 1998.

[278] Theopilos Papadimitriou, Konstantinos I. Diamantaras, Michael G. Strintzis, and Manos Roumeliotis, *Video scene segmentation using spatial contours and 3–d robust motion estimation*, IEEE Transactions on Circuits and Systems for Video Technology **14 (4)** (2004), 485–497.

[279] N. Papamarkos and B. Gatos, *A new approach for multilevel threshold selection*, CVGIP: Graphical Models and Image Processing **56(5)** (1994), 357–370.

[280] N. Papamarkos, C. Strouthopoulos, and I. Andreadis, *Multithresholding of color and gray–level images through a neural network technique*, Image and Vision Computing **18** (2000), 213–222.

[281] Ioannis Patras, Emile A. Hendriks, and Reginald L. Lagendijk, *Semi–automatic object–based video segmentation with labelling of color segments*, Signal Processing: Image Communication **18** (2003), 51–65.

[282] Fabien A. P. Petitcolas, *Watermarking schemes evaluation*, IEEE Signal Processing **17 (5)** (2000), 58–64.

[283] ———, *Stirmark benchmark 4.0*, www.petitcolas.net/ fabien/ watermarking/ evaluation/ index.html, Visited 2004.

[284] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn, *Attacks on copyright marking systems*, Information Hiding, Second International Workshop, IH98 (David Aucsmith, ed.), Proceedings, LNCS 1525, Springer–Verlag, Portland, Oregon, USA, 1998, pp. 219–239.

[285] Son Lam Phung, Abdesselam Bouzerdoum, and Douglas Chai, *A novel skin color model in ycbcr color space and its application to human face detection*, Proceedings of IEEE International Conference on Image Processing (ICIP2002), 2002, pp. 289–292.

[286] Son Lam Phung, Douglas Chai, and Abdesselam Bouzerdoum, *Adaptive skin segmentation in color images*, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (Hong Kong), 2003, pp. 173–176.

[287] Konstantinos N. Plataniotis and Anastasios N. Venetsanopoulos, *Color image processing and applications*, Springer–Verlag, Heidelberg–New York, 2000.

[288] Ilya Pollak, *Segmentation and restoration via nonlinear multiscale filtering*, IEEE Signal Processing Magazine **September** (2002), 26–36.

[289] C. A. Poynton, *Frequently asked questions about colour*, ftp://www.inforamp.net/ pub/ users/ poynton/ doc/ colour/ ColorFAQ.ps.gz., 1995.

[290] W.K. Pratt, *Spatial transform coding of color images*, IEEE Transaction on Communication Technology **19(6)** (1971), 980–992.

[291] J. M. S. Prewitt and M. L. Mendelsohn, *The analysis of cell images*, Annual New York Academy of Science **128** (1966), 1036–1053.

[292] R.J. Qian, M.I. Sezan, and K.E. Matthews, *A robust real-time face tracking algorithm*, Proceedings of IEEE International Conference on Image Processing, 1998, pp. 131–135.

[293] B. Qiu, V. Prinet, E.Perrier, and O. Monga, *Multi–block pca method for image change detection*, 12th International Conference on Image Analysis and Processing (ICIAP'03) (Mantova, Italy), 2003, p. 385.

[294] Francis K. H. Quek, Thomas Mysliwiec, and Meide Zhao, *Fingermouse: A freehand computer pointing interface*, Proceedings of the International Workshop on Automatic Face– and Gesture–Recognition (Zurich, Switzerland), 1995, pp. 372–377.

[295] D. Qzdemir and L. Akarun, *A fuzzy algorithm for color quantization of images*, Pattern Recognition **35** (2002), 17851791.

[296] Richard J. Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam, *Image change detection algorithms: A systematic survey*, IEEE Transactions on Image Processing (Submitted) (2004).

[297] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley, *Color transfer between images*, IEEE Computer Graphics and Applications **September/October** (2001).

[298] J. A. Richards, *Remote sensing digital image analysis*, Springer, Heidelberg, New york, 1993.

[299] Marty Rimm, *Marketing pornography on the information superhighway*, Georgetown Law Journal **83(5)** (1985), 1849–1934.

[300] Dirk Robinson and Peyman Milanfar, *Fundamental performance limits in image registration*, IEEE International Conference on Image Processing (ICIP'03), 2003.

[301] A. Rosenfeld and A. Kak, *Digital picture processing, vol. 2,* 2^{nd} *ed.*, Academic Press, New York, NY, 1982.

[302] Hazel Rossoti, *Colour: Why the world isn't grey*, Princeton University Press, Princeton, NJ, 1983.

[303] M. Roux, *Automatic registration of spot images and digitized maps*, Proceedings of the IEEE International Conference on Image Processing ICIP96 (Lausanne, Switzerland), 1996, pp. 625–628.

[304] J.J.K. Ruanaidh, W.J. Dowling, and F.M. Boland, *Watermarking digital images for copyright protection*, IEE Proceedings on Vision, Signal and Image Processing **143(4)** (1996), 250–256.

[305] D.L. Ruderman, T.W. Cronin, and C.C. Chiao, *Statistics of cone response to natural images: Implementation for visual coding*, Optical Doc. Of America **15(8)** (1998), 2036–2045.

[306] E.H. Ruspini, *A new approach to clustering*, Information Control **15** (1969), 22–32.

[307] J. C. Russ, *The image processing handbook*, CRC Press, Boca Raton, 1998.

[308] Mark A Ruzon and Carlo Tomasi, *Alpha estimation in natural images*, IEEE Conference on Computer Vision and Pattern Recognition (Hilton Head, South Carolina), 2000.

[309] Eli Saber, A. Murat Tekalp, and R. Eschbach andK. Knox, *Automatic image annotation using adaptive color classification*, Proceedings of Graphic Models and Image Processing, 1996, pp. 115–126.

[310] Eli Saber and A.M. Tekalp, *Frontal–view face detection and facial feature extraction using color, shape and symmetry based cost functions*, Pattern Recognition Letters **17(8)** (1998), 669–680.

[311] H. Samet, *Region representation: Quadtrees from boundary codes*, Comm. ACM **21** (1980), 163:170.

[312] B. Sankur, A. T. Abak, and U. Baris, *Assessment of thresholding algorithms for document processing*, Proceedings of the IEEE International Conference on Image Processing (Kobe, Japan), 1999, pp. 580–584.

[313] S. Santini and R. Jain, *Similarity measures*, IEEE Pattern Analysis and Machine Intelligence **21(9)** (1999).

[314] S. Satoh, Y. Nakamura, , and T. Kanade, *Name-it: Naming and detecting faces in news videos*, IEEE Multimedia **6(1)** (1999), 22–35.

[315] David Saxe and Richards Foulds, *Toward robust skin identification in video images*, Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996, pp. 379–384.

[316] R. Schettini, C. Brambilla, C. Cusano, and G. Ciocca, *On the detection of pornographic digital images*, Visual Communications and Image Processing **5150** (2003), 2105–2113.

[317] R. Schumeyer and K. Barner, *A color–based classifier for region identification in video*, Visual Communications and Image Processing **3309** (1998), 189–200.

[318] Shohreh Tabatabaii Seifi and Ali Qanavati, *Digital color image archive*, Qnavati@mehr.sharif.edu.

[319] M. Sester, H. Hild, and D. Fritsch, *Definition of ground control features for image registration using gis data*, Proceedings of the Symposium on Object Recognition and Scene Classification from Multispectral and Multisensor Pixels CD–ROM (Columbus, Ohio), 1998.

[320] A. Shahin, M. Menard, and M. Eboueya, *Cooperation of fuzzy segmentation operators for correction aliasing phenomenon in 3d color doppler imaging*, Artificial Intelligence **19(2)** (2000), 121154.

[321] B. Shiele and Alex Waibel, *Gaze based tracking based on face–color*, International Workshop on Automatic Face– and Gesture–Recognition (1995).

[322] Shih-Hao and Yuan-Pei Lin, *Wavelet tree quantization for copyright protection watermarking*, IEEE Transactions on Image Processing **13(2)** (2004), 154–165.

[323] Min C. Shin, Kyong I. Chang, and Leonid V. Tsap, *Does color space transformation make any difference on skin detection?*, IEEE Workshop on Applications of Computer Vision (Orlando, FL), 2002, pp. 275–279.

[324] E. Sifakis and G. Tziritas, *Moving object localization using a multi–label fast marching algorithm*, Signal Processing: Image Communication **16** (2001), 963–976.

[325] L. Sigal, S. Sclaroff, and V. Athitsos, *Estimation and prediction of evolving color distributions for skin segmentation under varying illumination*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2000, pp. 152–159.

[326] A. Singh, *Review article: Digital change detection techniques using remotely–sensed data*, International Journal of Remote Sensing **10(6)** (1989), 989–1003.

[327] W. Skarbek and A. Koschan, *Colour image segmentation-a survey*, Tech. report, Institute for Technical Informatics, Technical University of Berlin, 1994.

[328] J. Slater, *Modern television system to hdtv and beyond*, Pitman, London, 1991.

[329] Dan Smith, Richard Harvey, Yi Chan, and J. Andrew Bangham, *Classifying web pages by content*, IEE European Workshop on Distributed Imaging, vol. 99/109, 1999, pp. 8/1–8/7.

[330] J. R. Smith and S.-F. Chang, *Visual seek: A fully automated content–based image query system*, ACM Multimedia Conference, 1996.

[331] K. Sobottka and I. Pitas, *Face localization and feature extraction based on shape and color information*, Proceedings of the IEEE International Conference on Image Processing, 1996, pp. 483–486.

[332] Karin Sobottka and Ioannis Pitas, *Segmentation and tracking of faces in color images*, Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96), 1996, p. 236.

[333] ———, *Segmentation and tracking of faces in color images*, Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996, pp. 236–241.

[334] Hizir Sofyan, *Fuzzy clustering*, Statistical Case Studies (Wolfgang Hardle, Yuichi Mori, and Philippe Vieu, eds.), 2004, pp. 131–142.

[335] M. Soriano, S. Huovinen, B. Martinkauppi, and M. Laaksnoen, *Skin detection in video under changing illumination conditions*, Proceedings of the 15th International Conference on Pattern Recognition, 2000, pp. 839–842.

[336] S. Spors and R. Rabenstein, *A real–time face tracker for color video*, IEEE International Conference on Acoustics, Speech & Signal Processing (ICASSP) (Utah, USA), 2001.

[337] T. Starner and A. Pentland, *Real–time asl recognition from video using hmms*, Tech. Report 375, Media Lab, Massachusetts Institute of Technology, 1996.

[338] B Starykevitch and M Daoudi, *Poesia software architecture definition document*, www.poesia-filter.org/pdf/Deliverable_3_1.pdf, 2002.

[339] Helman Stern and Boris Efros, *Adaptive color space switching for face tracking in multi–colored lighting environments*, Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition (FGR'02), 2002.

[340] W.S. Stiles and Gunter Wyszecki, *Color science concepts and methods, quantitative data and formula*, John Wiley and Sons Inc., New York, 2000.

[341] Ahmad Subutai, *A usable real–time 3d hand tracker*, Conference Record of the Asilomar Conference on Signals, Systems and Computers, 1994, pp. 1257–1261.

[342] Q.B. Sun, W.M. Huang, , and J.K. Wu, *Face detection based on color and local symmetry information*, Proceedings of the Third International Conference on Automatic Face and Gesture Recognition, 1998, pp. 130–135.

[343] M. Swain and D. Ballard, *Color indexing.*, International Journal of Computer Vision **7(1)** (1991), 11–32.

[344] M.D. Swanson, M. Kobayashi, and A.H. Tewfik, *Multimedia data–embedding and watermarking technologies*, Proceedings of the IEEE, vol. 86(6), 1998, pp. 1064–1087.

[345] Daniel Sykora, *Inking old black and white cartoons*, Master's thesis, Department of Computer Science and Engineering, FEE, CTU, Prague, Czech Republic, 2003.

[346] Daniel Sykora and Jan Burianek Jiri Zara, *Unsupervised colorization of black–and–white cartoons*, NPAR'04.

[347] Chih-Wei Tang and Hsueh-Ming Hang, *A feature–based robust digital image watermarking scheme*, IEEE Transaction on Signal Processing **51(4)** (2003), 950–959.

[348] Ingeborg Tastl and Gunther Raidl, *Transforming an analytically defined color space to match psychophysically gained color distances*, the SPIE's 10th Int. Symposium on Electronic Imaging: Science and Technology (San Jose, CA), vol. 3300, 1998, pp. 98–106.

[349] May Taylor, Ethel Quayle, and Gemma Holland, *Child pornography, the internet and offending*, ISUMA - Canadian Journal of Policy Research **2(2)** (2001).

[350] J.M. Tenenbaum, T.D. Garvey, S. Weyl, and H.C. Welf, *An interactive facility for scene analysis research*, Tech. Report 87, Stanford Research Institute, AI Center, 1974.

[351] Jean-Christophe Terrillon, Martin David, and Shigeru Akamatsu, *Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments*, Proceedings of the 3rd. International Conference on Face & Gesture Recognition, 1998, p. 112.

[352] Jean-Christophe Terrillon, M. N. Shirazi, H. Sukamachi, and S. Akamatsu, *Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images*, Proceedings of the International Conference on Face and Gesture Recognition, 2000, pp. 54–61.

[353] J. Theiler and G. Gisler, *A contiguity–enhanced k–means clustering algorithm for unsupervised multispectral image segmentation*, Proc. SPIE, 1997, p. 108118.

[354] Naftali Tishby, Fernando C. Pereira, and William Bialek, *The information bottleneck method*, 37–th anuall Alerton Conference on Communication, Control , and Computing, 1999.

[355] F. Tomita, M. Yachida, and S. Tsuji, *Detection of homogenous regions by structural analysis*, Proceedings of the Joint Conference on Artificial Intelligence

(Stanford, CA), 1973, pp. 564–571.

[356] J. R. G. Townshend, C. O. Justice, and C. Gurney, *The impact of misregisitration on change detection*, IEEE Transaction on Geoscience and Remote Sensing **30(5)** (1992), 1054–1060.

[357] Peter Sean Tradile and Russel C. Conglaton, *A change–detection analysis: Using remotely sensed data to assess the progression of development in essex county, massachusetts from 1999 to 2001*, www.unh.edu/natural–resources/students.html.

[358] Ian Traynor, *Child porn verdict stuns net lawyers*, The Guardian (1998).

[359] M. Trivedi and J.C. Bezdek, *Low–level segmentation of aerial images with fuzzy clustering*, IEEE Transactions on Systems, Man, and Cybernetics **16(4)** (1986), 589598.

[360] Piyu Tsai, Yu-Chen Hu, and Chin-Chen Chang, *A color image watermarking scheme based on color quantization*, Signal Processing **84** (2004), 95–106.

[361] E.C.K. Tsao, J.C. Bezdek, and N.R. Pal, *Fuzzy kohonen clustering networks*, Pattern Recognition **27** (1994), 757–764.

[362] Yaakov Tsiag and Amir Averbuch, *Automatic segmentation of moving objects in video sequences: A region labelling approach*, IEEE Transaction on Circuits and Systems for Video Technology **12(7)** (2002), 597–612.

[363] Matthew Turk and Alex Pentland, *Eigenfaces for recognition*, Journal of Cognitive Neuroscience **3(1)** (1991), 71–86.

[364] G. Tziritas and C. Labit, *Motion analysis for image sequences coding*, Elsevier, Amsterdam, 1994.

[365] R. Venkatesh, K. R. Ramakrishnan, and S. H. Srinivasan, *Video object segmentation: A compressed domain approach*, IEEE Transactions on Circuits and Systems for Video Technology **14 (4)** (2004), 462–474.

[366] C. Vertan and N. Boujemaa, *Using fuzzy histograms and distances for color image retrieval*, Challenges of image retrieval (Brighton), 2000, p. 16.

[367] Luiz Filipe M. Vieira, Rafael D. Vilela, Erickson R. do Nascimento, Fernando A. Fernandes Jr., Rodrigo L. Carceroni, and Arnaldo de A. Araujo, *Automatically choosing source color images for coloring grayscale images*, Proceedings of the XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'03) (Sao Carlos, Brazil), 2003, p. 151.

[368] Vassili Sazonov Vladimir Vezhnevets and Alla Andreeva, *A survey on pixel-based skin color detection techniques*, GraphiCon'2003 (Moscow, Russia), 2003.

[369] Dan S. Wallach, *Copy protection technology is doomed*, IEEE Computer (2001), 48–49.

[370] C. Wang and M. Brandstein, *Multi–source face tracking with audio and visual data*, IEEE MMSP, 1999, pp. 169–174.

[371] James Ze Wang, Gio Wiederhold, and Oscar Firschein, *System for screening objectionable images using daubechies' wavelets and color histograms*, Computer Communications **21(25)** (1998), 1355–1360.

[372] Y. Wang and B. Yuan, *Fast method for face location and tracking by distributed behavior–based agents*, IEE Proceedings on Vision, Image, and Signal Processing **149(3)** (2002), 173–178.

[373] H. Wangand and S.-F. Chang, *A highly efficient system for automatic face region detection in mpeg video*, IEEE Trans. Circuits and Systems for Video Technology **7(4)** (1997), 615–628.

[374] Timothy Wark and Sridha Sridharan, *A syntactic approach to automatic lip feature extraction for speaker identification*, ICASSP, 1998, p. 36933696.

[375] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller, *Transferring color to grayscale images*, proceedings of ACM SIGGRAPH 2002 (San Antonio), 2002, pp. 277–280.

[376] O. Wenstop, *Motion detection from image information*, Proceedings of Scandinavian Conference on Image Analysis.

[377] J. S. Weska, *A survey of threshold selection techniques*, Computer Graphics and Image Processing **7(2)** (1978), 259–265.

[378] J. S. Weska, R. N. Nagel, and A. Rosenfeld, *A threshold selection technique*, IEEE Transaction on Computers **C–23 (12)** (1974), 1322–1326.

[379] Rafael Wiemker, Anja Spek, Daniel Kulbach, Hartwig Spitzer, and Hohann Bienlein, *Unsupervised robust change detection on multispectral imagery using spectral and spatial features*, Proceedings of the Third International Airborne Remote Sensing Conference and Exhibition (Copenhagen, Denmark), 1997.

[380] Paul Stefan Williams and Mike D. Alder, *Segmentation of natural images for cbir*, ICPR98, 1998.

[381] Chai Wah Wu, *On the design of content-based multimedia authentication systems*, IEEE Transactions on Multimedia **4 (3)** (2002), 385–393.

[382] Haiyuan Wu, Taro Yokoyama, Dadet Pramadihanto, and Masahiko Yachida, *Face and facial feature extraction from color images*, International Conference on Automatic Face and Gesture Recognition, 1996, pp. 345–350.

[383] Minghui Xia and Bede Liu, *Image registration by "super–curves"*, IEEE Transactions on Image Processing **13 No. 5** (2004), 720–732.

[384] Dong Xu, Xuelong Li, Zhenkai Liu, and Yuan Yuan, *Anchorperson extraction for picture in picture news video*, Pattern Recognition Letters **25** (2004), 1587–1594.

[385] M. Yamada, K. Ebihara, and J. Ohya, *A new robust method for extracting human silhouettes from color images*, Proceedings of the Third International Conference on Automatic Face and Gesture Recognition (Nara, Japan), 1998, pp. 528–533.

[386] Takahiro Yamamoto, Hiroshi Hanazumi, and Shinji Chino, *A change detection method for remotely sensed multi–spectral and multi–temporal images using 3–d segmentation*, IEEE Transaction on GRS **39(5)** (2001), 976–985.

[387] Wei-Qi Yan and Mohan S Kankanhalli, *Colorizing infrared home videos*, Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2003) (Baltimore), 2003.

[388] Yasuharu Yanamura and Hitoshi Saji, *Automatic registration of aerial image and digital map for detection of earthquake damaged area*, Proceedings of the VIIth Digital Image Computing: Techniques and Applications (Sydney), 2003, pp. 117–126.

[389] Jie Yang and Alex Waibel, *Tracking human faces in real–time*, Tech. Report CMU-CS-95-210, Carnegie Mellon University, 1995.

[390] ———, *A real–time face tracker*, Proceedings of the Third Workshop Applications of Computer Vision, 1996, pp. 142–147.

[391] M. Yang and N. Ahuja, *Gaussian mixture model for human skin color and its application in image and video databases*, Proceedings of the SPIE: Conf. on Storage and Retrieval for Image and Video Databases (SPIE 99), 1999, pp. 458–466.

[392] Ming-Hsuan Yang and Narendra Ahuja, *Detecting human faces in color images*, Proceedings of the IEEE International Conference on Image Processing, 1998, pp. 127–130.

[393] Ming-Hsuan Yang, David J. Kriegman, and Marenda Ahuja, *Detecting faces in images: A survey*, IEEE Transactions on Pattern Analysis ad Machine Intelligence **24(1)** (2002), 34–58.

[394] Mehran Yazdi and Andre Zaccarin, *Interframe coding using deformable triangles of variable size*, ICIP'97, 1997, pp. 456–459.

[395] Qixiang Ye, Wen Gao, and Wei Zeng, *Color image segmentation using density–based clustering*, Proceedings of 2003 IEEE International Conference on Acoustic and Signal Processing (Hog Kong), 2003, pp. 401–404.

[396] Lijun Yin, Jingrong Jia, and Joseph Morrissey, *Towards race–related face identification: Research on skin color transfer*, Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition (FGR'03), 2003.

[397] Seong-Joon Yoo, *Intelligent multimedia information retrieval for identifying and rating adult images*, Lecture Notes in Artificial Intelligence **3213** (2004), 164–170.

[398] Seong-Joon Yoo, Min ho Jung, Hee Beom Kang, Chee Sun Won, and Soo-Mi Choi, *Composition of mpeg-7 visual descriptors for detecting adult images on the internet*, Lecture Notes in Computer Science **2713** (2003), 682–687.

[399] Pao-Ta Yu, Hung-Hsu Tasi, and Juh-Shyan Lin, *Digital watermarking based on neural networks for color images*, Signal Processing **81** (2001), 663–671.

[400] Young-Dal Yu, Dae-Seong Kang, and Daijin Kim, *Color image compression based on vector quantization using pca and lebd*, IEEE IENCON'99, 1999, pp. 1259–1262.

[401] A.L Yuille, D. Snow, and M. Nitzberg, *Signfinder : Using color to detect, localize and identify informational signs*, Int. Conf. on Computer Vision ICCV98, 1998, pp. 629–633.

[402] B. D. Zarit, B. J. Super, and Francis K. H. Quek, *Comparison of five color models in skin pixel classification*, ICCV99 Intl Workshop on recognition, analysis and tracking of faces and gestures in Real–Time systems, 1999, pp. 58–63.

[403] Benjamin D. Zarit, *Skin detection in video images*, Tech. Report VISLab-99-01, VISLab, University of Illinois at Chicago.

[404] ———, *Skin detection in video images*, Master's thesis, University of Illinois at Chicago, Chicago, 1999.

[405] Benjamin D. Zarit and Boaz J. Super Francis K.H. Quek, *Comparison of five color models in skin pixel classification*, Tech. Report VISLab-99-07, University of Illinois at Chicago.

[406] H. Zheng, M. Daoudi, and B. Jedynak, *Adult image detection using statistical model*, Compression et Reprsentation des Signaux Audiovisuels (Lille, France), 2004.

[407] ———, *Adult image filtering for web safety*, Proceedings of 2nd International Symposium on Image/Video Communications Over Fixed and Mobile Networks (Brest, France), 2004, pp. 77–80.

[408] H. Zheng, M. Daoudi, C. Tombelle, and C. Djeraba, *Adult image filtering for internet safety*, Multimedia Security Handbook (B. Fuhrt and D. Kirouski, eds.), CRC Press, 2004.

[409] H. Zheng, H. Liu, and M. Daoudi, *Blocking objectionable images: adult images and harmful symbols*, Proceedings of IEEE International Conference on Multimedia and Expo (Taipei, Taiwan), 2004.

[410] Huicheng Zheng, *Maximum entropy modelling for skin detection: with an application to internet filtering*, Ph.D. thesis, University of Lille, France, 2004.

[411] Huicheng Zheng, Mohamed Daoudi, and Bruno Jedynak, *Blocking adult images based on statistical skin detection*, Electronic Letters on Computer Vision and Image Analysis **4(2)** (2004), 1–14.

[412] Qing-Fang Zheng, Wei Zeng, Gao Wen, and Wei-Qiang Wang, *Shape–based adult image detection*, Proceedings of the Third International Conference on Image and Graphics, 2004, pp. 150–153.

[413] Francesco Ziliani and Andrea Cavallaro, *Image analysis for video surveillance based on spatial regularization of a statistical model–based change detection*, Real–Time Imaging **7** (2001), 389–399.

[414] Barbara Zitova and Jan Flusser, *Image registration methods: A survey*, Image and Vision Computing **21** (2003), 977–1000.

[415] Thomas Zoller, Lothar Hermes, and Joachim M. Buhmann, *Combined color and texture segmentation by parametric distributional clustering*, Proceedings of ICPR, vol. 2, 2002, pp. 627–630.