Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression

John R. Smith

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Graduate School of Arts and Sciences

Columbia University

1997

ABSTRACT

Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression

John R. Smith

This thesis presents new methods for integrating spatial and feature information in order to improve systems for image retrieval, analysis and compression. In particular, this thesis develops and demonstrates an integrated spatial and feature query paradigm for image retrieval, a general framework for extracting spatially localized features from images using histogram and binary set back-projections, a system for image compression and analysis based upon a joint-adaptive space and frequency (JASF) graph, and a representation of texture based upon spatial-frequency energy histograms.

These techniques are similar in their exploitation of images as two-dimensional, non-stationary signals. The visually important information within images is often confined to spatially localized regions, or is represented by the spatial arrangements of these regions. By developing processes that analyze and represent images in this way, we improve our capabilities to develop powerful content-based image retrieval and image compression systems.

This thesis examines several image system applications, such as image and video storage and retrieval, content-based image query, adaptive image compression and image analysis. We demonstrate that by using the proposed integrated spatial and feature approaches, we improve image system performance in each application over non-integrated methods.

Contents

1 Introduction and Motivation **1**
1.1. Visual information retrieval . . . 1
1.1.1 Why unconstrained imagery? . . . 1
1.1.2 Image and video storage and retrieval systems . . . 2
1.2. Problem definitions . . . 3
1.2.1 Content-based visual query . . . 3
1.2.2 Spatial image query . . . 5
1.2.3 Spatial and feature image query . . . 6
1.2.4 Proposed solutions . . . 7
1.3. Summary of contributions . . . 8
1.4. Outline of thesis . . . 8

2 Representing Color **10**
2.1. Introduction . . . 10
2.2. Feature representation . . . 10
2.3. Related work on color image retrieval . . . 10
2.4. Colorimetry . . . 11
2.5. Color image perception . . . 11
2.6. Color space transformation and quantization . . . 11
2.6.1 Color transformation . . . 12
2.6.2 Color quantization . . . 12
2.7. Color spaces . . . 13
2.7.1 *RGB* color space . . . 13
2.7.2 Linear transform color spaces . . . 13
2.7.3 Munsell color order system . . . 14
2.7.4 CIE color spaces . . . 15
2.7.5 HSV color space . . . 15
2.8. Representing color image features . . . 17
2.8.1 Color histograms . . . 17
2.8.2 Color sets . . . 17
2.8.3 Color histogram compaction . . . 18
2.8.4 Binary set fast map (BSFM) . . . 18
2.9. Summary . . . 20

3 Representing Texture **21**
3.1. Introduction . . . 21
3.2. What is texture? . . . 21
3.3. Related work on texture . . . 23
3.4. Related work on texture image retrieval . . . 23
3.5. Representations of texture . . . 24
3.5.1 Space/spatial-frequency expansion . . . 24
3.5.2 Gabor Functions . . . 24
3.5.3 Wavelet expansion . . . 26
3.5.4 QMF filter bank (FB) . . . 26
3.5.5 Tree structured separable QMF FB . . . 27

3.5.6 Representing texture by *s/s-f* energies . . . 29
3.6. Texture classification experiments . . . 30
3.6.1 K-Class discrimination . . . 30
3.6.2 Classification results and discussion . . . 31
3.7. Texture element histogram . . . 35
3.7.1 Texture filter bank (FB) . . . 35
3.7.2 Texture channel generator (TCG) . . . 35
3.7.3 Texture space transformations . . . 37
3.7.4 Texture channel quantization . . . 39
3.7.5 Texture histograms . . . 43
3.7.6 Texture sets . . . 43
3.7.7 Texture set example . . . 43
3.8. Summary . . . 45

4 Feature Similarity and Indexing **46**
4.1. Introduction . . . 46
4.2. Histogram space . . . 46
4.2.1 Histogram normalization . . . 46
4.2.2 Histogram metric space . . . 47
4.3. Taxonomy of distance metrics . . . 47
4.3.1 Minkowski-form distance . . . 47
4.3.2 Quadratic-form distance . . . 51
4.3.3 Non-histogram distance . . . 52
4.3.4 Distance metric comparison . . . 53
4.4. Retrieval effectiveness evaluation . . . 53
4.4.1 Image test set . . . 53
4.4.2 Retrieval effectiveness . . . 54
4.4.3 Color retrieval . . . 54
4.4.4 Texture retrieval . . . 56
4.4.5 Color and texture retrieval . . . 57
4.5. Feature indexing . . . 58
4.5.1 Prefiltering . . . 58
4.5.2 Binary set bounding (BSB) . . . 59
4.5.3 Dimensionality reduction . . . 59
4.5.4 Query optimized distance (QOD) computation . . . 60
4.6. Summary . . . 62

5 Region Extraction **82**
5.1. Introduction . . . 82
5.2. Visual feature extraction . . . 82
5.2.1 Region extraction strategies . . . 83
5.2.2 Pattern recognition . . . 84
5.2.3 Image segmentation . . . 85
5.2.4 Region detection . . . 85
5.3. Back-projection algorithms . . . 88
5.3.1 Confidence back-projection ($\mathcal{BP}1$) . . . 88
5.3.2 Quadratic confidence back-projection ($\mathcal{BP}2$) . . . 89
5.3.3 Local histogramming ($\mathcal{BP}3$) . . . 89
5.3.4 Binary set back-projection ($\mathcal{BP}4$) . . . 89
5.3.5 Single element quadratic back-projection ($\mathcal{BP}5$) . . . 89
5.4. Back-projection region extraction examples . . . 89
5.4.1 Color histogram back-projection . . . 89
5.4.2 Object detection and histogram back-projection . . . 90
5.4.3 Texture histogram back-projection . . . 90
5.4.4 Combining color and texture histogram back-projection . . . 90
5.5. Region extraction systems . . . 90

5.5.1 Single-element quadratic back-projection system . . . 93
5.5.2 Binary set iteration (BSI) system . . . 93
5.6. Summary . . . 93

6 Adaptive Image Analysis and Compression **95**
6.1. Introduction . . . 95
6.1.1 Signal expansion . . . 95
6.1.2 Adaptive signal expansions . . . 96
6.1.3 Outline . . . 98
6.2. Image signal expansion . . . 98
6.2.1 Series expansion . . . 98
6.2.2 Filter bank . . . 98
6.2.3 Finite signal expansion . . . 99
6.2.4 Partitionable expansions . . . 100
6.2.5 Image expansion . . . 103
6.3. Image segmentation . . . 103
6.3.1 Binary segmentation . . . 103
6.3.2 Image spatial quadtree (QT) segmentation . . . 104
6.4. Combining frequency expansion and segmentation . . . 104
6.4.1 Non-partitionable expansion cascade . . . 104
6.4.2 Partitionable expansion cascade . . . 105
6.4.3 Two-step expansion cascades . . . 105
6.4.4 Multiple step expansion cascades . . . 107
6.5. Tree and graph structured expansions . . . 107
6.5.1 Complexity . . . 108
6.5.2 Single-trees . . . 108
6.5.3 Double-tree (DT) . . . 109
6.5.4 Dual double tree (DDT) . . . 109
6.5.5 Redundant space and frequency tree (RSFT) . . . 111
6.5.6 Joint adaptive space and frequency (JASF) graph . . . 111
6.6. Basis element library . . . 113
6.6.1 Basis sets . . . 113
6.6.2 Basis selection . . . 114
6.6.3 Rate-distortion optimization . . . 114
6.7. Image compression examples . . . 115
6.7.1 Haar filter JASF graph . . . 115
6.7.2 QMF JASF graph . . . 116
6.8. Summary . . . 116

7 Spatial and Feature Query **117**
7.1. Introduction . . . 117
7.1.1 Integrating spatial and feature query . . . 117
7.2. Region query . . . 118
7.2.1 Absolute spatial location . . . 118
7.2.2 Size . . . 120
7.2.3 Region query strategies . . . 120
7.3. Multiple regions query . . . 122
7.3.1 Multiple regions query strategy – absolute locations . . . 123
7.3.2 Region relative location . . . 123
7.3.3 Spatial indexing . . . 125
7.3.4 Special spatial relations . . . 127
7.3.5 Multiple regions query strategy – relative locations . . . 127
7.3.6 Symbolic image retrieval . . . 128
7.3.7 Relative spatial query efficiency . . . 128
7.4. Feature query . . . 130
7.4.1 Histogram quadratic distance . . . 130

7.4.2 Color sets . . . 130
7.4.3 Color region extraction . . . 131
7.4.4 Color set query strategy . . . 131
7.4.5 Synthetic color region image retrieval . . . 132
7.4.6 Color photographic image retrieval . . . 135
7.5. Summary . . . 139

8 Content-based visual query applications **141**
8.1. Introduction . . . 141
8.2. WebSEEk image and video search engine . . . 141
8.2.1 Overview of search engines . . . 141
8.2.2 Image and video collection process . . . 142
8.2.3 Subject classification process . . . 144
8.2.4 Search and retrieval processes . . . 147
8.2.5 Content-based techniques . . . 149
8.2.6 Evaluation . . . 151
8.3. VisualSEEk color/spatial image query system . . . 155
8.3.1 System design . . . 155
8.3.2 Relevance feedback queries . . . 156
8.3.3 Image annotation . . . 159
8.3.4 Text-based searching . . . 159
8.3.5 Color/spatial image query . . . 159
8.4. SaFe spatial and feature query system . . . 159
8.4.1 System design . . . 159
8.4.2 Image query process . . . 159
8.4.3 SaFe image query examples . . . 160
8.5. Summary . . . 160

9 Conclusion and future directions **164**
9.1. Conclusion . . . 164
9.2. Future directions . . . 165
9.3. Afterword . . . 165

References **166**

Appendix **177**
A. 1-Partitionable frequency expansion . . . 177
B. Partitionable Haar filterbank . . . 178

List of Figures

1-1 Image and video storage and retrieval. . . . 2
1-2 Taxonomy of image query methods: CBIQ = content-based image query, CBRQ = content-based region query, SQ = spatial query, abs SQ = absolute spatial query, rel SQ = relative spatial query, SaFe = spatial and feature image query. . . . 3
1-3 Content-based image query (CBIQ). . . . 4
1-4 Spatial image query, (a) query image (b) possible absolute spatial match, (c) possible relative spatial match. . . . 6
1-5 Integrated spatial and feature query. . . . 7
1-6 Map outlining the work addressed in this thesis. . . . 9

2-1 Transformation (T_c) and quantization (Q_c) produce the color feature space in which the color histograms $\mathbf{h}_c$ and binary color sets $\mathbf{s}_c$ are defined. . . . 13
2-2 Transformation T_c from RGB to HSV and quantization gives 18 *hues*, 3 *saturations*, 3 *values* and 4 *grays* = 166 colors. . . . 17
2-3 Relationship between 3-D RGB color histogram and binary color set. . . . 18
2-4 Compaction of color histogram energy for regions vs. images. . . . 19

3-1 Example textures from the Brodatz texture collection: d001: Woven aluminum wire, d056: Straw matting, d095: Brick wall, d020: French canvas, d014: Woven aluminum wire, d006:Woven aluminum wire, d003: Reptile skin, d004: Pressed cork, d087: Sea fan, fossilized with a coral covering, d005: Expanded mica, d111: Plastic bubbles, d066: Plastic pellets, d011: Homespun woolen cloth, d103: Loose burlap, d049: Straw screening, d015: Straw. . . . 22
3-2 Image formation from texture surface. . . . 23
3-3 Gabor filter spectrum where contours indicate the half-peak magnitudes of filters responses. The Gabor filter responses to pure sinusoids: (a) $\theta = 0$ and $\omega = 30$ and (c) $\theta = 45$ and $\omega = 60$, are depicted in (b). . . . 25
3-4 2-D QMF filter bank (FB), (a) *s-f* plane partitioning, (b) subband interpretation. . . . 26
3-5 QMF wavelet FB responses to pure sinusoids: (a) $\theta = 0$ and $\omega = 30$, (b) wavelet image for (a), and (c) $\theta = 45$ and $\omega = 60$, (d) wavelet image for (c). . . . 27
3-6 QMF filters (a) low pass filter h_0, (b) high-pass filter h_1. . . . 27
3-7 Magnitude of the frequency responses H_0 and H_1. . . . 28
3-8 2-D wavelet QMF FB. . . . 28
3-9 Wavelet transformation of a composite image of Brodatz textures. . . . 29
3-10 Brodatz texture classification experiment, texture cuts. . . . 32
3-11 Brodatz texture classification experiment, retrieved texture cuts that best match query texture cut. . . . 32
3-12 Effect on classification rate of the class size. . . . 33
3-13 Effect on classification rate of the number of training classes. . . . 33
3-14 Effect on classification rate of the number of discriminant functions used for classification. . . . 34
3-15 Effect on classification rate of the number of feature set. . . . 34
3-16 Generation of texture feature set. The filter bank (FB) and texture channel generator (TCG) generate nine texture channels. Transformation T_t and quantization Q_t produce the texture feature space in which texture histograms $\mathbf{h}_t$ and binary texture sets $\mathbf{s}_t$ are defined. . . . 35
3-17 Texture filter bank (FB) and texture channel generator (TCG). . . . 36
3-18 Overall process for representing texture using texture histograms. . . . 36

3-19 Rotated texture energy features: (a) several rotations of Brodatz texture d049 Straw screening, (b) wavelet images (c) L_2 subband energy values. . . . 38
3-20 Scaled texture energy features: (a) several scalings of the Brodatz texture d035 Lizard skin, (b) wavelet images (c) L_2 subband energy values. . . . 40
3-21 The quantizer Q produces output $\hat{x}$ for each input x. The quantization error is $\hat{x} - x$. . . . 41
3-22 Texture channel quantizer training process. . . . 42
3-23 Determination of optimal $J = 2$ output quantizer for texture channel $k = 1$. . . . 42
3-24 Determination of optimal $J = 2$ output quantizer for texture channel $k = 8$. . . . 43
3-25 Mean texture histograms from 2240 Brodatz texture cuts (a) obtained using Max-Lloyd trained texture channel quantizers, (b) obtained using median trained texture channel quantizers. . . 44
3-26 Mean binary texture sets from 2240 Brodatz texture cuts (a) obtained using Max-Lloyd trained texture channel quantizers, (b) obtained using median trained texture channel quantizers. . . 44
3-27 Example of texture set representation using a $k = 3$ channel FB. . . . 44

4-1 Feature vector representation of discrete-space, two-bin normalized histograms ($r = 1$). . . . 47
4-2 Feature vector representation of discrete-space, two-bin normalized histograms ($r = 2$). . . . 47
4-3 Minkowski-form distance metrics compare only "like" bins between the histograms. . . . 48
4-4 The cosine distance metric considers the angle θ between vectors, irrespective of vector lengths. The euclidean distance metric $\mathcal{D}2$ considers vector lengths and angle between them. . . . 50
4-5 Binary set Hamming distance metric ($\mathcal{D}3$) computes the exclusive OR between elements. The '$\star$'s indicate the positions of bit difference. . . . 50
4-6 Quadratic-form distance metrics compare multiple bins between the histograms using similarity matrix $A = [a_{ij}]$. . . . 51
4-7 Distance metric retrieval effectiveness evaluation experiment setup. . . . 53
4-8 Results of histogram dimensionality reduction: "A" = DCT, "B" = SVD, "C" = PCA, "D" = HS (optimal sorting). . . . 60
4-9 Average retrieval effectiveness for 23 queries for sunset images using QOD query method, K166 = all 166 colors, K6 = best 6 colors, K4 = best 4 colors, K2 = best 2 colors, K1 = best color. 61
4-10 Query 1(a). Average retrieval effectiveness for 23 queries for sunset images (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$). . . . 63
4-11 Query 1(b). Average retrieval effectiveness for 23 queries for sunset images (distances $\mathcal{D}5$, $\mathcal{D}6$, $\mathcal{D}7$, $\mathcal{D}8$). . . . 64
4-12 Query 2(a). Average retrieval effectiveness for 21 queries for pink flowers images (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$). . . . 65
4-13 Query 2(b). Average retrieval effectiveness for 21 queries for pink flowers images (distances $\mathcal{D}5$, $\mathcal{D}6$, $\mathcal{D}7$, $\mathcal{D}8$). . . . 66
4-14 Query 3(a). Average retrieval effectiveness for 42 queries for blue sky images (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$). . . . 67
4-15 Query 3(b). Average retrieval effectiveness for 42 queries for blue sky images (distances $\mathcal{D}5$, $\mathcal{D}6$, $\mathcal{D}7$, $\mathcal{D}8$). . . . 68
4-16 Query 4(a). Average retrieval effectiveness for 41 queries for lion images (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$). . . . 69
4-17 Query 4(b). Average retrieval effectiveness for 41 queries for lion images (distances $\mathcal{D}5$, $\mathcal{D}6$, $\mathcal{D}7$, $\mathcal{D}8$). . . . 70
4-18 Query 5.1(a). Average retrieval effectiveness for 20 queries for Brodatz texture (Brodatz #D4: Pressed cork) (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$, $\mathcal{D}5$). . . . 71
4-19 Query 5.2(a). Average retrieval effectiveness for 20 queries for Brodatz texture (Brodatz #D1: Woven aluminum wire) (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$, $\mathcal{D}5$). . . . 72
4-20 Query 6(a). Average retrieval effectiveness for 189 queries for coral images (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$). . . . 73
4-21 Query 6(b). Average retrieval effectiveness for 189 queries for coral images (distances $\mathcal{D}5$, $\mathcal{D}6$, $\mathcal{D}7$, $\mathcal{D}8$). . . . 74
4-22 Query 6(c). Average retrieval effectiveness for 189 queries for coral images using color *vs.* texture (distances texture: T($\mathcal{D}2$), texture: T($\mathcal{D}4$), color: C($\mathcal{D}2$), color: C($\mathcal{D}4$)). . . . 75
4-23 Query 7(a). Average retrieval effectiveness for 26 queries for wildcats images (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$). . . . 76

4-24 Query 7(b). Average retrieval effectiveness for 26 queries for wildcats images (distances $\mathcal{D}5$, $\mathcal{D}6$, $\mathcal{D}7$, $\mathcal{D}8$). 77
4-25 Query 7(c). Average retrieval effectiveness for 26 queries for wildcats images using color *vs.* texture (distances texture: T($\mathcal{D}2$), texture: T($\mathcal{D}4$), color: C($\mathcal{D}2$), color: C($\mathcal{D}4$)). 78
4-26 Query 8(a). Average retrieval effectiveness for 34 queries for yellow flowers images (distances $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$). 79
4-27 Query 8(b). Average retrieval effectiveness for 34 queries for yellow flowers images (distances $\mathcal{D}5$, $\mathcal{D}6$, $\mathcal{D}7$, $\mathcal{D}8$). 80
4-28 Query 8(c). Average retrieval effectiveness for 34 queries for yellow flowers images using color *vs.* texture (distances texture: T($\mathcal{D}2$), texture: T($\mathcal{D}4$), color: C($\mathcal{D}2$), color: C($\mathcal{D}4$)). 81

5-1 Strategy for region extraction in the integrated spatial and feature image query system. . . . 82
5-2 Pattern recognition is the process of classifying image points into a set of known classes (Ω). (a) Brodatz texture composite image, (b) example of labels (ω_k) assigned by pattern recognition process. 84
5-3 Image segmentation is the process of segmenting the image into non-overlapping regions. (a) Brodatz texture composite image, (b) example results of image segmentation. 85
5-4 Demonstration of multiple solutions for image segmentation (a) unsegmented image, (b) segmentation into six regions (b) segmentation into eleven regions. 86
5-5 Region detection is the process of detecting a model pattern within an image. (a) Brodatz texture composite image, (b) example of the region detected for model pattern = "Cane." . . 86
5-6 Color histogram back-projection comparing methods $\mathcal{BP}1$, $\mathcal{BP}2$, $\mathcal{BP}3$, and $\mathcal{BP}4$ in detection of yellow grass. 90
5-7 Color histogram back-projection comparing methods $\mathcal{BP}1$, $\mathcal{BP}2$, $\mathcal{BP}3$, and $\mathcal{BP}4$ in "Find Waldo". 91
5-8 Texture histogram back-projection comparing methods $\mathcal{BP}1$, $\mathcal{BP}2$, $\mathcal{BP}3$, and $\mathcal{BP}4$ in detection of zebra stripes. 92
5-9 Combined color/texture back-projection. 92
5-10 Results of the single color quadratic back-projection system ($\mathcal{BP}5$) for automated region extraction from the "Baboon" image. 94
5-11 Results of the single color quadratic back-projection system ($\mathcal{BP}5$) for automated region extraction from the "Stop sign" image. 94

6-1 Example time-frequency bases for a four point 1-D signal. The squares depict the 18 possible ways to partition the time-frequency plane. The vertical lines correspond to segmentations in time while the horizontal lines correspond to segmentations in frequency: s = segmentation-only, w = wavelet packet (WP)-tree, d = double-tree (DT), g = JASF graph. 96
6-2 (a) Wavelet packet (WP)-tree adapts to global frequency, (b) spatial quadtree (QT) adapts to spatial content. (White lines = frequency expansion, black lines = spatial segmentation) . . . 97
6-3 (a) Double-tree (DT) expansion treats space and frequency asymmetrically, (b) JASF graph expansion treats space and frequency symmetrically. (White lines = frequency expansion, black lines = spatial segmentation) 97
6-4 Two-channel filter bank (FB). Analysis filtering with $\mathbf{H}_i$ and subsampling generate subbands Y_0 and Y_1. Upsampling and synthesis filtering with $\mathbf{G}_i$ and adding reconstruct $\tilde{X} = X$. . . . 99
6-5 Binary segmentation and resynthesis. 104
6-6 Hybrid cascades with frequency expansion and segmentation ($F^{(1)} = \{\mathbf{H}_0^{(1)}, \mathbf{H}_1^{(1)}\}$, and $(F^{(1)})^{-1} = \{\mathbf{G}_0^{(1)}, \mathbf{G}_1^{(1)}\}$). 106
6-7 Hybrid cascade case 1: $F^{(1)} \rightarrow S \rightarrow (F^{(1)})^{-1} \rightarrow S^{-1}$. 106
6-8 Hybrid cascade case 2: $S \rightarrow F^{(1)} \rightarrow S^{-1} \rightarrow (F^{(1)})^{-1}$. 106
6-9 K step expansion cascade in space and one step in frequency. 107
6-10 JASF graph combines k-PFE filter banks ($F^k = \{\mathbf{H}_0^{(k)}$ and $\mathbf{H}_1^{(k)}\}$) and segmentation (S). . . 107
6-11 Using compact notation, (a) WP-tree, and (b) spatial QT, where the numbers $(1, 4, 16, \ldots)$ give the number of nodes (or basis elements) at each level. The number of nodes will be omitted in later figures. 108
6-12 Tree-structured filter bank (FB). 109
6-13 Tree-structured segmentation. 110
6-14 Asymmetric joint space and frequency expansions (a) DT, (b) DDT 110

6-15 Redundant space and frequency tree (RSFT) provides a symmetric expansion in segmentation and frequency operations. . . . 111
6-16 Example RSFT expansion of an image. In the RSFT, the basis elements generated by the $F \rightarrow S$ branch are approximately equivalent to those generated by the $S \rightarrow F$ branch. . . . 112
6-17 The JASF graph provides symmetric expansion in space and frequency, and avoids the "near" redundancy of the RSFT. . . . 112
6-18 Selection of node $t_{1,1,0,1}$ excludes all darkened nodes from the basis. . . . 113
6-19 Example of basis selection which cannot be obtained via the DT or WP-tree. (a) A basis from dyadic time/frequency library, (b) corresponding time/frequency tiling. . . . 114
6-20 Rate-distortion operating points (R_i, D_i) and optimal R-D point selection via trade-off constraint λ. . . . 115
6-21 JASF compression, (a) JASF graph basis for *Barbara* image, (b) reconstructed at 0.25 bpp. . . . 116

7-1 Matches are found for each query region q_k. Target images T that contain matches to all query regions are candidate image matches. . . . 117
7-2 Region spatial distance (a) fixed spatial distance $d^s_{q,t}$, (b) bounded spatial distance $d^{s'}_{q,t}$, where R^{xy} determines the valid spatial bounds. . . . 119
7-3 Spatial indexing techniques, (a) the spatial quad-tree indexes region centroids, and (b) the r-tree indexes region MBRs (rectangles) by using overlapping buckets. . . . 119
7-4 Parallel attribute query strategy for a single region query with attributes of feature value, location, area and spatial extent. . . . 121
7-5 Pipeline attribute query strategy for a single region query with attributes of feature value, location, area and spatial extent. . . . 122
7-6 Overall strategy for computing image matches by joining individual regions queries. . . . 123
7-7 Symbolic image retrieval – absolute spatial query matches. . . . 124
7-8 2-D string representation of spatial relationships (a) un-rotated 2-D string, (b) rotated projection. 124
7-9 2-D String projection. . . . 125
7-10 2-D string compare pseudo-code. . . . 126
7-11 Symbolic image retrieval – relative spatial query matches. . . . 126
7-12 Symbolic image retrieval – "nearness" relative spatial query matches. . . . 127
7-13 2-D strings rotation invariance (a) 0° ($A < C < B, ABC$), (b) 90° ($ABC, A < C < B$), (c) 45° ($A < C < B, ABC$) with projection onto the rotated axis. . . . 128
7-14 Symbolic image retrieval – absolute spatial query matches. . . . 128
7-15 SaFe user interface with returned symbolic images. . . . 129
7-16 Relative spatial query (a) query response time *vs.* image database size, (b) query response time *vs.* number of query symbols. . . . 130
7-17 Example of the absolute location synthetic image queries: query image is top left, retrieved images from top left to bottom right in order of best match. . . . 133
7-18 Example of the relative location synthetic image queries: query image is top left, retrieved images from top left to bottom right in order of best match. . . . 133
7-19 Generation of synthetic images and ground truth database and process for evaluating effectiveness of joint spatial and color queries. . . . 134
7-20 Average retrieval effectiveness of 100 randomly generated queries on database of synthetic images, with the four query methods defined as above. . . . 135
7-21 SaFe user interface to the photographic image query system. . . . 136
7-22 Query 1. Sunset image retrievals ($\mathcal{A}$ = SaFe query, $\mathcal{B}$ = color histograms, $\mathcal{C}$ = color sets). . . . 137
7-23 Query 2. Blue sky with greenery image retrievals ($\mathcal{A}$ = SaFe query, $\mathcal{B}$ = color histograms, $\mathcal{C}$ = color sets). . . . 138
7-24 Example queries (a) region with absolute location, (b) two regions with absolute locations, (c) multiple regions with relative locations, (d) multiple regions with both absolute and relative locations. . . . 139

8-1 Image and video gathering process. . . . 143
8-2 The *Traversal Spider* traverses the Web and assembles the list of *URL*s of images and videos. 143
8-3 Portion of the image and video subject taxonomy $\{\widehat{s_m}\}$. . . . 146
8-4 Search, retrieval and search results list manipulation processes. . . . 147

8-5 Search results for `SUBJECT` = "nature." 148
8-6 Content-based visual query results for images/videos $\simeq$ "red race car". 150
8-7 WebSEEk utilizes the SaFe integrated spatial and feature image search tools. SaFe finds the images which contain similar regions in similar spatial arrangements to the query. 152
8-8 Relevance feedback search allows user to select both positive and negative examples. 153
8-9 VisualSEEk client-server architecture. 155
8-10 Data flow between client and server applications in the VisualSEEk system. 156
8-11 An example VisualSEEk query that specifies (a) three color regions - blue, green and light blue, and their locations and sizes, and (b) the query results. 157
8-12 Relevance feedback query. (a) Images selected at random from the database, and (b) the results of a color-query using the image in (a) third down on the left. 157
8-13 Image annotation process. (a) Individual image annotation. (b) "Power" annotation allows groups of similar images to be annotated at once. 158
8-14 Text and color-searching. (a) The text-based search using keyword "lion" yields only two matches. (b) The color search using one lion image from (a) finds many more images of lions. 158
8-15 SaFe system architecture. Major subsystems are UI = user-interface, IA = image analysis subsystem, QE = query engine and DB = image database. 160
8-16 SaFe image retrieval of synthetic color-region images. 161
8-17 SaFe image retrieval of "unconstrained" color photographs. 162

List of Tables

2.1 Comparisons of color transforms (L = linear, NL = non-linear). 17

3.1 Rotated texture features. Energy is roughly constant within the same scale during rotation. . 37
3.2 Scaled texture features. 39

4.1 Summary of the eight histogram distance metrics ($\mathcal{D}1 \ldots \mathcal{D}8$). 48
4.2 Query 1: Sunsets. Comparison of eight distance metrics. 55
4.3 Query 2: Flowers. Comparison of eight distance metrics. 55
4.4 Query 3: Nature with blue sky. Comparison of eight distance metrics. 55
4.5 Query 4: Lions. Comparison of eight distance metrics. 56
4.6 Query 5.1: Brodatz texture (Brodatz #D4: Pressed cork). Comparison of five distance metrics. 56
4.7 Query 5.2: Brodatz texture (Brodatz #D1: Woven aluminum wire). Comparison of five distance metrics. 57

6.1 Compression results on the *Barbara* image using Haar filter. 115
6.2 Compression results on the *Barbara* image using QMF12a filter. 116

7.1 The `REGION` table for query image $Q = \{q_0, q_1\}$. 118
7.2 The `REGION` table for target image $T = \{t_0, t_1, t_2, t_3, t_4\}$. 118
7.3 The `REGION` relation with attributes for features $\mathbf{f}$, region centroids (x, y), region sizes `area` and widths and heights (w, h) of the minimum bounding rectangles (MBRs). 121
7.4 The `COLORSET` relation with attributes for the decomposed quadratic distance equation parameters μ_t and $r_t[m]$'s. Denotation by * indicates that a secondary index is built on the attribute in order to allow range queries to be performed on that attribute. 131

8.1 MIME mapping between extensions and object types. 144
8.2 Sample (a) terms and their counts $\{t_k : f_k\}$ and (b) key-terms counts $\{t_k^* : f_k\}$ with subjects s_m's and mappings $\mathcal{M}_{km} : t_k^* \rightarrow s_m$'s. Taken from the assessment of over $500,000$ images and videos. 146
8.3 Rates of correct subject classification (precision) for random set of classes. 154
8.4 Rates of correct automated type classification. 154

List of Abbreviations

BSB	binary set bounding
BSFM	binary set fast map
BSI	binary set iteration
CBVQ	content-based visual query
CIE	Commission Internationale de l'Eclairage
DCT	discrete cosine transform
DDT	dual double-tree
DT	double-tree
FB	filter bank
FDA	Fisher discriminant analysis
HBP	histogram back projection
HS	histogram sorting
HSV	hue-saturation-value color space
HTML	hypertext markup language
HTTP	hypertext transfer protocol
IVSR	image and video storage and retrieval
JASF	joint adaptive space and frequency
JPEG	Joint Photography Experts Group
MPEG	Motion Picture Experts Group
PCA	principal component analysis
QBIC	query by image content
QMF	quadrature mirror filter
QOD	query optimized distance
QT	quad-tree
PR	perfect reconstruction
RGB	red-green-blue color space
RSFT	redundant space and frequency tree
SaFe	spatial and feature
S/S-F	space/spatial-frequency
TCG	texture channel generator
WP	wavelet packets
WWW	World-Wide Web

Chapter 1

Introduction and Motivation

1.1. Visual information retrieval

The tremendous growth in the numbers and sizes of digital image and video collections is making necessary the development of tools for indexing this unconstrained imagery. In order to provide a system with image and video search capabilities, content-based visual query (CBVQ) techniques are being developed which index the visual features of the images [102, 3, 112, 105, 156, 146] and videos [153, 45].

The objectives of CBVQ are, in the absence of image understanding, to obtain and utilize discriminants that are useful in conducting similarity queries for visual information. Recent efforts of CBVQ have focused on a few specific visual dimensions, such as color, texture, shape, motion and spatial information. However, without integrating these visual dimensions, the current content-based techniques have limited capacity to characterize the content of the visual imagery and to satisfactorily retrieve images.

This thesis presents a powerful enhancement of the recent CBVQ techniques which integrates spatial and feature information in order to improve systems for image retrieval, analysis and compression. In particular, we develop several new processes that include:

1. **Analysis** – a technique for extracting spatially localized color and texture regions from images using histogram and binary set back-projection,
2. **Retrieval** – a system for integrating spatial and feature querying of image databases, and
3. **Compression** – a new data structure and algorithm for analyzing and compressing images, which is based upon a joint-adaptive space and frequency graph.

These techniques are similar in their exploitation of images as two-dimensional, non-stationary signals. The visually important information within images is often contained within spatially localized regions, or is represented by the overall existence and spatial arrangements of these regions. By developing processes that analyze and represent images in this way, we improve our capabilities to develop powerful content-based image retrieval and compression systems. This thesis presents several new techniques for integrated spatial and feature image systems and demonstrates improved performance in various applications.

1.1.1 Why unconstrained imagery?

The type of visual information referred to in this work is considered to be "unconstrained" because it is not restricted to a particular domain of imagery. As such, we do not utilize any of the domain-specific constraints, which are often components of traditional pattern recognition solutions [40]. The development of systems that analyze, compress and retrieve unconstrained imagery is extremely relevant due to the enormous amount of this type of imagery that is accessible and is accumulating.

Consider that everyday we view various forms of visual media, such as television broadcasts, photographs, graphics, animations and videos. These media are provided and stored increasingly in digital form. For example, the World-Wide Web (WWW) is one such source for viewing hundreds of gigabytes of digital visual information [143]. While there is great accessibility to large stores of digital imagery, new systems need to be developed to better manage, index and search for the visual information.

Since there is no restriction on the content of this visual information, we develop a class of techniques for image systems which does not depend on the content or domain of imagery. However, the techniques we propose are applicable in a variety of particular domains of imagery such as satellite images [28, 87, 82, 92],

medical images [83, 79, 108, 135], environmental images [105] and geographical images [16, 15, 21]. In these domains, representations of color and texture that are analogous to those for photographic images do exist. Furthermore, the advantages in providing techniques for spatial and feature analysis of the type defined in this thesis are also clear.

1.1.2 Image and video storage and retrieval systems

We formulate the basic problem that is addressed in this thesis as the search and retrieval of unconstrained images and videos by using visual features. We consider the framework of the typical image and video storage and retrieval (IVSR) environment, which is depicted in Figure 1-1.

We assume that the images and videos are stored in a digital form – in a database, digital archive, file system or distributed network. We consider collections that contain on the order of 100 to 10,000,000 images and videos. In the search and retrieval environment, users located on a network (i.e., local area network, intranet, internet, WWW, or dial-up) search the collections to find particular images and videos. Several example applications of IVSR include:

- stock photography distribution [3],
- Web-based image and video search engines [148, 147, 47, 70],
- on-line publication of multimedia documents [173, 88, 64],
- video-on demand (VOD) [130, 38, 88, 19],
- video editing [96, 95] and video re-purposing [37],
- on-line news, on-line sports and other types of photo-journalism, and so forth.

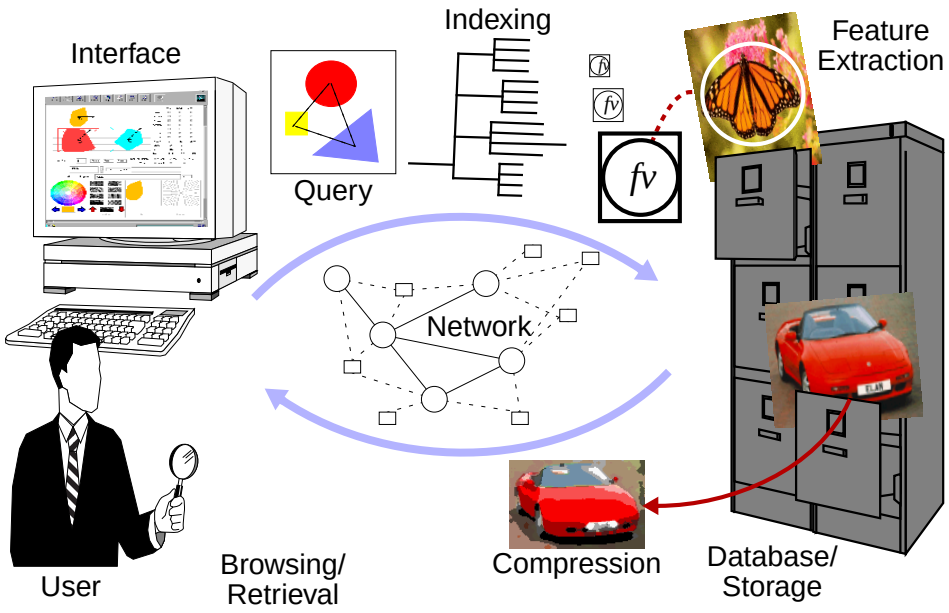


Figure 1-1: Image and video storage and retrieval.

In a straight-forward implementation of the picture database management system (PDBMS), text, annotations, and other attributes and records allow the users to search using information related to the pictures. For example, textual and keyword annotations may include descriptions of the subjects of images, or a photographer's name, date, and so forth. This meta-data is often stored and accessed by using traditional relational database management systems (RDBMS) or object-oriented databases (OODB) (for example, [21, 16, 15, 161, 28]).

However, this approach provides for limited capacity for retrieving visual information. For one, the associated meta-data cannot fully describe the contents of the imagery. Very often, the images and videos

are annotated manually, which is extremely time-consuming and human-resource intensive, especially if the archive is large. In general, this approach is not scalable to large archives. Furthermore, it does not anticipate the myriad of ways in which the content of the visual information is of interest to the users.

To enhance this approach, content-based visual query (CBVQ) techniques use the visual features of the images and videos in the search process. The CBVQ systems extract and index visual features such as colors [159, 53, 102, 158, 66, 157], textures [117, 51, 92, 118], shapes [72, 135, 169, 85] and motions [137, 13]. The user constructs visual queries using graphical interface tools (for example [71, 151, 142]), or provides examples of images and videos in order to search for and retrieve the desired items from the archive [3, 45].

Other important components of the system include: image and video compression for efficiency in storage, browsing and transmission [149], feature access [93, 17], progressive retrieval [109] and viewing [123], and the use of icons [22, 37, 162, 148]. The systems may also provide tools for browsing and navigating through the archive using the visual features [177] and for learning from interaction with the user [97]. For example, relevance feedback techniques [7, 77] guide the system and the user to the desired visual information.

1.2. Problem definitions

The techniques developed in this thesis provide several strategies for computing image queries that specify the features, sizes and arbitrary spatial layouts of regions, which include both absolute and relative spatial locations. We also address several special case spatial queries involving adjacency, overlap and encapsulation of regions. Figure 1-2 summarizes the relationship of these query methods. We demonstrate that the integrated spatial and feature approach improves image query capabilities over non-spatial, content-based approaches.

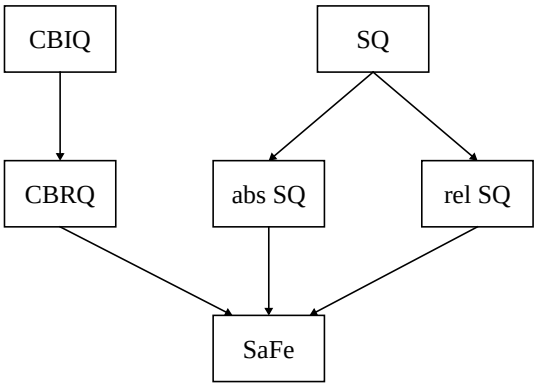


Figure 1-2: Taxonomy of image query methods: CBIQ = content-based image query, CBRQ = content-based region query, SQ = spatial query, abs SQ = absolute spatial query, rel SQ = relative spatial query, SaFe = spatial and feature image query.

1.2.1 Content-based visual query

The objective of content-based visual query (CBVQ) is to efficiently find and retrieve those images from a database that are most similar to the user's query image. CBVQ is different from a typical database query in that it entails a similarity search. There are two forms of CBVQ:

- K-nearest neighbor query, which retrieves the K images that are most similar to the query image, and
- range query, which retrieves all images that are within a fixed similarity bound, $\mathcal{D}_f \leq \tau_f$, from the query image.

The difference is subtle, but it impacts both the strategies for computing queries (see Section 4.5.1) and the interaction with the user. We further distinguish between two uses of CBVQ: content-based image query (CBIQ) and content-based region query (CBRQ).

1.2.1.1 Content-based image query

Problem 1 Content-based image query (CBIQ). *K-nearest neighbor query. Given a collection $\mathcal{C}$ of N images and a feature dissimilarity function $\mathcal{D}_f$, find the K images $\mathcal{I}_T \in \mathcal{C}$ with the lowest dissimilarity, $\mathcal{D}_f(\mathcal{I}_Q, \mathcal{I}_T)$, to the query image $\mathcal{I}_Q$.*

In this case, a query always returns K images, which are typically sorted by lowest dissimilarity to the query image. The query computation needs to be more thorough than the bounded CBVQ in order to consider images that are outside of the similarity bounds in providing the K matches.

Problem 2 Bounded content-based image query. *Range query. Given a collection $\mathcal{C}$ of N images and a feature dissimilarity function $\mathcal{D}_f$, find the images $\mathcal{I}_T \in \mathcal{C}$ such that $\mathcal{D}_f(\mathcal{I}_Q, \mathcal{I}_T) \leq \tau_f$, where $\mathcal{I}_Q$ is the query image and τ_f is a threshold of feature similarity.*

In this case, a query returns any number of images depending on the bounds defined by the threshold of feature similarity, τ_f. The formulation of CBVQ as a bounded similarity search is particularly important in the investigation of techniques for speeding up the search process. For example, the criteria for evaluating query speed-up techniques include the requirement that no images from within the similarity bounds are lost in the search process (no false dismissals, see Section 4.5.1).

We explore both types of CBVQ: unbounded and bounded similarities searches. We note that for any unbounded query there is an equivalent bounded query with some dissimilarity threshold, which returns only the top K matches. As such, we will only differentiate between the two formulations when necessary.

In order to provide for fast comparison of images, as illustrated in Figure 1-3, the CBVQ system typically extracts, stores and indexes the features from the images off-line. For example, color histograms, which describe the distribution of colors in an image, are pre-computed, stored and used as a basis for measuring the similarity of images in [45, 3, 145].

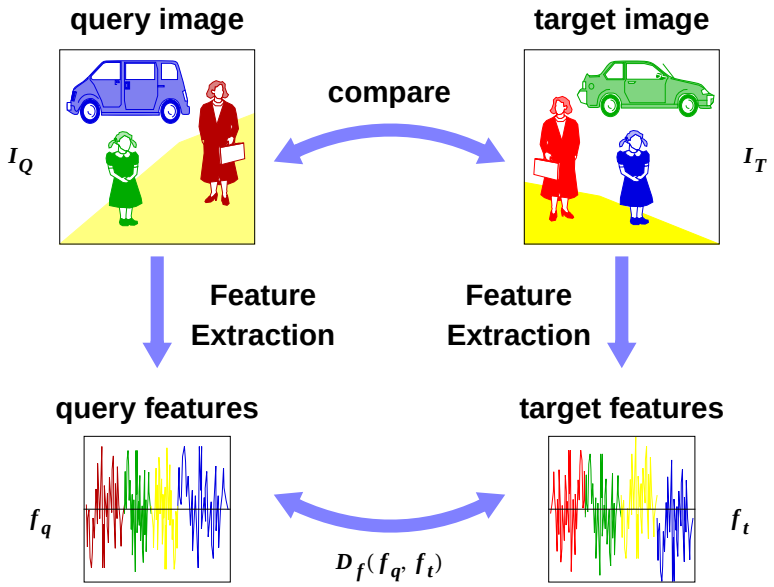


Figure 1-3: Content-based image query (CBIQ).

The dissimilarity metric is devised to produce small values for similar images and large values for dissimilar images. However, the performance of this method is limited by the quality of the feature extraction. Color histograms and other similar global measures of image "content" provide insufficient information for comparing images.

The process of CBVQ has been investigated in several recent efforts: IBM's QBIC project [102], MIT's Photobook system [112], the Chabot image retrieval system [105], and Virage's commercial image retrieval system [3]. These systems use feature-based approaches for indexing visual information. QBIC has investigated features such as color, texture and shape. Photobook has investigated texture, shape and facial features.

QBIC has placed a primary research focus on devising strategies for indexing the features [59]. On the other hand, Photobook has investigated new methods for representing image features which not only provide for discrimination of images, but also preserve their semantic information.

When the image database is large and the image feature representations are complex, the exhaustive search of the database and computation of the image similarities is not expedient. Several recent techniques have been proposed to speed-up image retrieval. Swain and Ballard [159], Stricker and Orengo [157] and QBIC [102] pre-compute and utilize only simple and fairly compact image features. Petrakis and Faloutsos [114] and QBIC [44] use a combination of data reduction techniques and efficient indexing structures, one such structure being the R-tree [58], to speed-up image retrieval. QBIC [59] has also investigated effective pre-filtering techniques [59] for reducing the amount of computations at query time. This thesis also presents two new techniques for reducing query computation: binary set bounding (BSB) and query optimized distance (QOD) computation.

However, recent approaches for CBVQ have neglected an important criteria for similarity: region and spatial information and spatial relationships.

1.2.1.2 Content-based region query (CBRQ)

In a content-based region query (CBRQ), the images are compared on the basis of their regions. This differs from the previous formulation of CBVQ in Problem 1, which uses global feature measurements from the images. However, the CBRQ can be viewed as an extension of CBVQ in that a first stage of the query computes the CBVQ on regions instead of images. Then, the final stage of the query determines the images corresponding to the regions and computes the overall image distance by weighting the region distances.

Problem 3 Content-based Region Query (CBRQ). *Given a collection $\mathcal{C}$ of N images and feature dissimilarity function $\mathcal{D}_f$, find the K best images $\mathcal{I}_T \in \mathcal{C}$ that have at least R regions such that $\mathcal{D}_f(\mathcal{I}_Q^R, \mathcal{I}_T^R) \leq \tau_f$, where $\mathcal{I}_Q^R$ is the query image with R regions and τ_f is a threshold of feature similarity.*

CBRQ is improved by adding spatial information into the query. As such, the overall dissimilarity measure considers the feature and spatial values of the regions. First, we examine the problem of spatial image query.

1.2.2 Spatial image query

In an effort largely different from CBVQ, researchers have developed techniques for spatial indexing which retrieve images based on the locations of objects [55, 22, 27, 58, 56, 113]. These approaches compare symbolic images in which the regions or objects have been defined *a priori*, as depicted in Figure 1-4. Spatial indexing does not incorporate features such as color, texture, shape, and so forth. It is difficult to extend spatial indexing to include feature measures of the objects [154].

We distinguish between two types of spatial indexing: relative and absolute. In relative spatial indexing, images are matched based upon the relative locations of symbols. For example, a relative spatial query may ask for images in which symbol A is to the left of symbol B. In absolute spatial indexing, images are matched based upon fixed positions in the images. For example, an absolute spatial query may ask for images in which symbol A is in the upper left quadrant and symbol B is on the bottom of the image. The two types of spatial image query are depicted in Figure 1-5. An example query image is given in Figure 1-5(a). The image in Figure 1-5(c) satisfies the relative spatial requirements of the symbols in the query. The image in Figure 1-5(b) does not satisfy relative position constraints, but the symbols are close to the absolute positions specified in the query image.

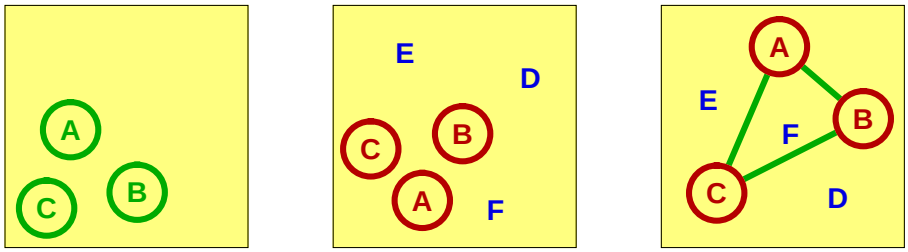
1.2.2.1 Relative spatial image query

The relative spatial image query is defined as follows:

Problem 4 Relative Spatial Query. *Given a collection $\mathcal{C}$ of N symbolic images, find the images $\mathcal{I}_T \in \mathcal{C}$ that contain at least R symbols in the same spatial arrangement as the R symbols in the example (query) symbolic image.*

1.2.2.2 Absolute spatial image query

The absolute spatial image query is defined as follows:



(a) (b) (c)

Figure 1-4: Spatial image query, (a) query image (b) possible absolute spatial match, (c) possible relative spatial match.

Problem 5 Absolute Spatial Query. *Given a collection $\mathcal{C}$ of N symbolic images and spatial distance function $\mathcal{D}_s$, find the K best images $\mathcal{I}_T \in \mathcal{C}$ that have at least R regions such that $\mathcal{D}_s(\mathcal{I}_Q^R, \mathcal{I}_T^R) \leq \tau_s$, where $\mathcal{I}_Q^R$ is the query image with R regions and τ_s is a threshold of spatial distance.*

1.2.3 Spatial and feature image query

We introduce a more powerful type of image search system that allows users to flexibly query for images by specifying both the visual features and the spatial properties of the desired images. The integrated spatial and feature querying gives the user control in selecting the regions and attributes that are most important in determining similarity in a given query. As such, the system accommodates partial matching of images as determined by the user.

1.2.3.1 Absolute spatial and feature query

The absolute spatial and feature query process integrates absolute spatial querying and content-based region querying. A recent approach by Stricker and Dimai [157] investigates one form of absolute spatial and feature querying. Their system divides each image into five fuzzy regions and extracts a color moment feature set for each. In this way, the system provides for querying by the five absolute region locations. However, it is not possible to query for images by specifying either arbitrary regions or the spatial relationships of regions.

Problem 6 Absolute Spatial and Feature Query. *Absolute Spatial Query (Problem 5) + CBRQ (Problem 3).*

1.2.3.2 Relative spatial and feature query

The relative spatial and feature query process integrates content-based region querying and relative spatial querying.

Problem 7 Relative Spatial and Feature Query. *CBRQ (Problem 3) + Relative Spatial Query (Problem 4).*

1.2.3.3 Integrated spatial and feature query

We demonstrate in Chapter 7 that integrated spatial and feature image querying provides a powerful method for image retrieval. However, it is extremely complex in that it requires that several disparate image query techniques be combined. First, the feature query component requires the assessment of feature similarities of regions. Second, the spatial query component requires the assessment of the similarities in spatial locations and sizes of regions. Finally, the system requires that the spatial relationships, such as "above," "below," and so forth, be resolved. We demonstrate an image query system which solves these problems and integrates the separate methods into a single system.

Problem 8 Integrated Spatial and Feature Query (SaFe). *Absolute Spatial Query (Problem 6) + CBRQ (Problem 3) + Relative Spatial Query (Problem 4).*

In the integrated spatial and feature query system, regions and their feature and spatial attributes are first extracted from the images. The comparison of the images then considers the similarities of the feature and spatial attributes of the regions as depicted in Figure 1-5. The overall match score between images is computed by summing the weighted distances between the best matching regions in terms of spatial locations, sizes and features. The relative spatial locations of regions in the target images are also compared to those in the query image to determine the image matches.

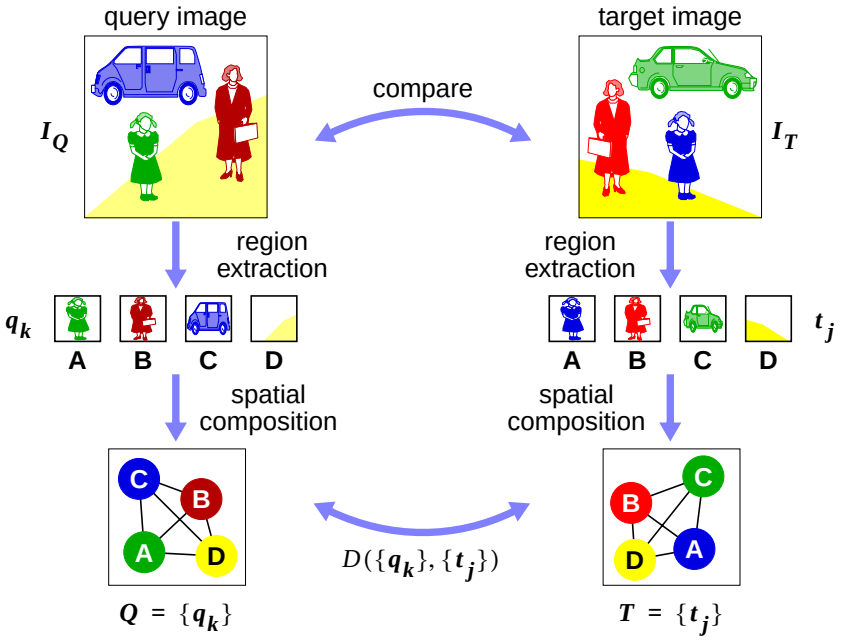


Figure 1-5: Integrated spatial and feature query.

1.2.3.4 Survey and related work

Recently developed image retrieval systems (for example, [102, 112, 105, 3]) have not integrated spatial and feature querying methods. The QBIC system [102] provides querying of whole images (CBIQ) and manually extracted regions (CBRQ) by color, texture and shape but not by spatial relationships. The Virage system [3] provides for querying of only image global features such as color, composition, texture and structure. Pass, Zabih and Miller [110] devised an image retrieval system which considers the spatial adjacency of colors. Their technique splits the global image histogram into coherent and scattered components. The measure of color coherence identifies the existence of connected colored regions. Although the technique improves on color histogram indexing, it does not support querying by the spatial locations of the color regions.

Jacobs, Finkelstein and Salesin [71] devised a system which uses spatial information and visual features represented by dominant wavelet coefficients. Their system allows the user to sketch example images and provides for improved matching over image distance norms. However, their technique provides for little flexibility in specifying approximate and relative spatial information.

1.2.4 Proposed solutions

In order to fully integrate CBVQ and spatial image query capabilities, we devise an image similarity function which contains both spatial and feature components. We first note that the perceived attributes of an image consist of both region and image attributes. For example, the region attributes refer to region features, sizes and spatial locations. The image attributes refer to the number of regions and their spatial arrangement.

In order to quickly process queries, we design the representations for the region attributes such as the region features, spatial location and size to require minimal computation in matching. The region attributes

are indexed directly to allow for maximal efficiency in queries. The overall image matches are determined from the candidate regions retrieved in the region queries.

In this way, a query specified by the user is translated directly into pruning operations on the region attributes. The image attributes, such as region relative locations and special spatial relations, are resolved only in the final stage of the query. This is because these evaluations have the highest complexity. The pruning performed by the queries on the regions reduces the number of candidate images and regions that need to be evaluated at the final stages.

1.3. Summary of contributions

We summarize the contributions of this thesis in improving systems for retrieving, analyzing and compressing images as follows: in this thesis we

1. demonstrate a new integrated spatial and feature query paradigm for image retrieval,
2. develop a new representation of texture based upon texture channel energy histograms, which is symmetric to that for color histograms,
3. explore new binary set representations of color and texture and their applications in region extraction and integrated spatial and feature image retrieval,
4. present two new approaches, binary set bounding (BSB) and query-optimized distance (QOD) computation, for efficiently computing feature similarity queries,
5. develop a new general framework for extracting spatially localized features from images using histogram and binary set back-projections,
6. develop a new system for image compression and analysis based upon a joint-adaptive space and frequency (JASF) graph, which includes:
   (a) the development of the theory of partitionable expansions,
   (b) the creation of a new framework for developing general tree- and graph-structured signal decompositions, which include spatial quad-tree, wavelet packets, the double-tree and the JASF graph,
7. implement several new image and video retrieval prototype systems, which provide for
   (a) cataloging of and searching for visual information on the World-Wide Web,
   (b) content-based and relevance feedback searching for images, and
   (c) integrated spatial and feature image querying.

1.4. Outline of thesis

The thesis is organized as follows (see Figure 1-6): we develop representations of color in Chapter 2 and texture in Chapter 3. In both visual dimensions, we first obtain multiple channel representations of images. Through processes of transformation and quantization of the channels, we obtain feature spaces for measuring the color and texture elements, respectively. From these, we obtain the histograms and binary sets. In short, a color histogram defines the distribution of colors and a texture histogram defines a distribution of texture elements. The binary sets define only the selection of colors and texture elements, respectively.

In Chapter 4, we define metrics for discriminating among images using measures of the similarities of color and texture, respectively. We provide a thorough evaluation of eight distance metrics along the dimensions of (1) retrieval effectiveness in example image queries and (2) query computation efficiency. We also introduce the two powerful techniques for computing the feature queries: binary set bounding (BSB) and query optimized distance (QOD) computation.

In Chapter 5, we investigate the process of extracting color and texture regions from imagery. We present the histogram back-projection (HBP) procedure for detecting and extracting color and texture regions. We describe and compare the performance of five back-projection algorithms and present some examples of color and texture back-projection. We also develop a procedure for automatically extracting regions from images by using a binary set iteration (BSI) and back-projection procedure.

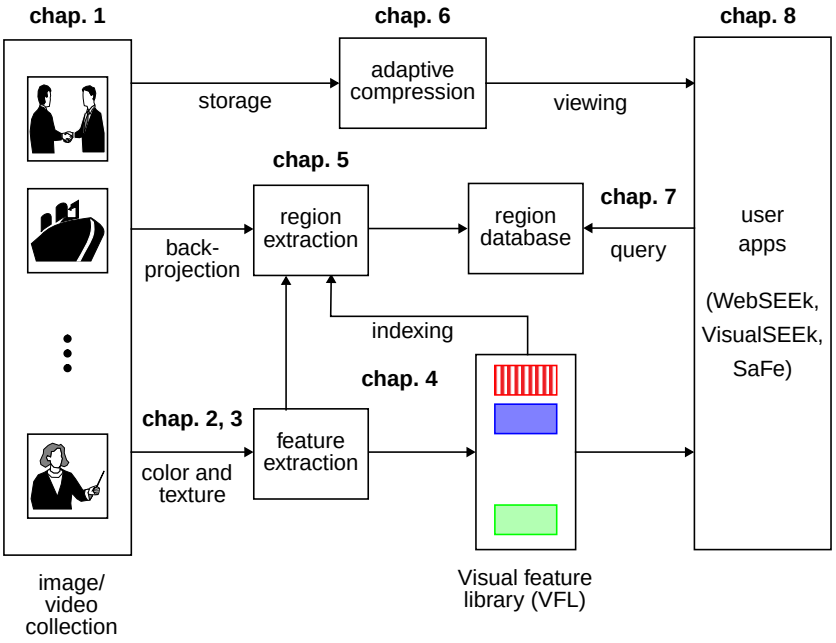


Figure 1-6: Map outlining the work addressed in this thesis.

In Chapter 6, we present a new system adaptive image compression and image analysis. The joint adaptive space and frequency (JASF) graph integrates image frequency expansion and segmentation, symmetrically, to achieve a joint expansion. Using the framework of partitionable expansions, we develop the highly efficient and powerful JASF graph-based expansion of images. We demonstrate that the JASF graph improves access to image features, and improves image compression performance over several recent image-adaptive compression techniques.

In Chapter 7, we present new prototype systems for computing integrated spatial and feature image queries. We describe the query processes and index structures used to facilitate the search for images. We demonstrate that the spatial and feature paradigm improves image retrieval effectiveness over non-spatial content-based approaches.

Finally, in Chapter 8 we present our recently developed systems for visual information retrieval that demonstrate the image search concepts derived in this thesis: WebSEEk, VisualSEEk, and SaFe. We show that the integrated spatial (relative and absolute locations) and feature (color and texture) query systems provide a powerful solution to problems of image and video search and retrieval. We also demonstrate that these techniques are easily integrated into diverse applications such as one for cataloging images and videos from the World-Wide Web, and another for integrating visual features and text in the classification, search and retrieval of images.

Chapter 2

Representing Color

2.1. Introduction

We present a unified framework for representing the color and texture features. The objectives are to provide a basis for

1. assessing the similarities of the visual features of images,
2. storing and indexing the visual feature information and
3. extracting the salient regions from images.

We will show that these objectives are accomplished by representing color and texture by histograms. The histograms provide a measure of the distribution of features inside a region or image. By appropriately designing the feature spaces in which the histograms are defined, the histograms sufficiently represent the visual dimensions of color and texture and satisfy the above objectives.

In this chapter (for color), and the next chapter (Chapter 3, for texture), we generate and evaluate the feature spaces for color and texture which allow for representation of color and texture using histograms and binary sets. For both visual dimensions, we show that candidate feature spaces are generated through the processes of transformation T and quantization Q of the image data. The design goal is to identify the transformation and quantization pairs: (T_c, Q_c) for color, and (T_t, Q_t) for texture that generate perceptually uniform, complete and compact feature spaces. We also examine the special case of histograms that are binary. These binary sets are extremely compact and are suitable for representing, storing, indexing (see Chapter 4) and extracting image features (see Chapter 5).

2.2. Feature representation

In both visual dimensions, color and texture, the image points are initially represented as multi-dimensional feature points: a *color point* is defined by values in three color channels and a *texture point* is defined by the energy in nine spatial-frequency (*s-f*) channels. The feature points are transformed and quantized to generate feature spaces with 166 *colors* and 512 *texture elements*, respectively.

Using these feature elements, we utilize two approaches each for representing *color* and *texture*: histograms and binary sets. Histograms are a distribution of feature elements. Binary feature sets are a selection of feature elements.

2.3. Related work on color image retrieval

The use of color histograms for image retrieval has been previously explored in [159, 103, 145, 43, 110]. The fundamental elements of this approach include the selection of the color feature space (T_c, Q_c) and the histogram distance metric $\mathcal{D}_c$. There has been no consensus about which color feature space is most suitable for color histogram-based image retrieval. The problem is a result of the fact there does not exist a universally accepted color space, and color perception is significantly subjective [175]. Therefore, we have seen a large variety of color spaces used in practice.

For example, Swain and Ballard [159] utilize an opponent-axis (OPP) color system [4] quantized to 2048 colors in a system for color image retrieval. The IBM QBIC system [103] first quantizes the RGB color space

into 4096 colors (16 levels per each color channel). Then, each color is transformed to the Munsell color space by the MTM transform [98]. Finally, clustering determines the k best Munsell colors (typically, $k = 64$).

On the other hand, Pass, Zabih and Miller [110] simply utilize the RGB color space, quantized uniformly to 64 colors, in their image retrieval system. Finally, Gray [53], to retrieve images, transforms RGB first to CIE-LUV and then partitions the CIE-LUV color space to 512 colors.

In this chapter, we present a framework for selecting and evaluating the color spaces for the purpose of color histogram-based retrieval. We begin by discussing the basic properties of color and colorimetry.

2.4. Colorimetry

Electromagnetic radiation $F(\lambda)$ in the range of visible light ($\lambda \in \{380\text{nm}\ldots780\text{nm}\}$) is perceived as color or colored light. It has been verified experimentally that color is perceived through three independent color receptors which have peak response at approximately, red, green and blue wavelengths, $\lambda_r = 700\text{nm}, \lambda_g = 546.1\text{nm}, \lambda_b = 435.8\text{nm}$, respectively. By assigning each primary color receptor, $k \in \{r, b, g\}$, a response function $c_k(\lambda)$, visible light of any color $F(\lambda)$, is represented by a linear superposition of the $c_k(\lambda)$'s [104], as follows: by normalizing $c_k(\lambda)$'s to reference white light $W(\lambda)$ such that

$$W(\lambda) = \overline{c}_r(\lambda) + \overline{c}_g(\lambda) + \overline{c}_b(\lambda),$$

$F(\lambda)$ produces the tristimulus responses (R, G, B), such that

$$F(\lambda) = R\,\overline{c}_r(\lambda) + G\,\overline{c}_g(\lambda) + B\,\overline{c}_b(\lambda). \tag{2.1}$$

As such, any color can be represented by a linear combination of the three primary colors (R, G, B).

2.5. Color image perception

The perception of color images is a complex process, and we make several simplifying assumptions: we assume that color is a single pixel process. In this sense, the perception of color is assumed to not be influenced by surrounding colors. We also assume that viewing conditions, such as ambient lighting, viewer adaptation, viewing distance, and image display quality can be ignored. Generally, it is difficult to control these parameters in the diverse application environments that are anticipated for color image retrieval.

2.6. Color space transformation and quantization

The space spanned by the R, G, and B values (see Eq 2.1) is complete in that all colors are represented as vectors in the 3-D RGB space. We use this space as the starting point for representing color features in images. We investigate the transformations T_c of RGB and subsequent quantizations Q_c that generate partitioned color spaces with properties of uniformity, completeness, compactness and naturalness. These spaces allow for image color content to be defined by color histograms and color sets.

Property 1 Uniformity. *The metric proximity between colors indicates the perceived similarity of colors.*

We will see in Chapter 3 that the color-based image comparison utilizes a proximity metric of the color features. For one, this comparison requires that color proximity is simple to compute. One way to assure this, is to design the color space transformation such that color proximity is not a function of the position in color space. The transformation T_c, which is the primary determinant of the uniformity of the color space, also determines along with quantization Q_c, it's completeness and compactness.

Property 2 Completeness. *The color space includes all perceptually distinct colors.*

It is necessary that the color space is complete in that all perceptually distinct colors are included. Visual completeness does not guarantee mathematical completeness, however, the converse is true. In general, if the transformation T_c is invertible, then the un-quantized color space will be complete. In this case, the quantization Q_c will determine the completeness.

Property 3 Compactness. *Each color in the color space is perceptually distinct from the other colors.*

In order to restrict the dimensionality of the image color feature representation, or similarly, the number of colors, the color space is non-redundant or "compact." Mathematically, nonredundancy is assured when the mapping from RGB is one-to-one or many-to-one. Visually, nonredundancy requires that no two colors are so similar that they are perceived as identical. In general, selection of quantization Q_c will involve a trade-off between completeness and compactness.

Property 4 Naturalness. *The color space provides for a natural breakdown of colors into the three basic perceptual attributes of color: brightness, hue and saturation.*

The separation of color into components of brightness, hue and saturation are intuitive for the user [68]. The ease in navigating the color space impacts the user's ability to construct color-based image queries.

1. Brightness. Attribute of visual sensation which corresponds to a color exhibiting more or less light. Variations range from bright to dim.
2. Hue. Attribute of visual sensation which corresponds to a color appearing similar to one or two of the perceived colors red, yellow, green and blue.
3. Saturation. Attribute of visual sensation which corresponds to a color exhibiting more or less of its hue. Saturation allows the judgment to be made of the degree to which a colored light differs from an achromatic light regardless of their brightness.

We design the transformation T_c and quantization Q_c to produce a color space that meets these objectives. These conditions are typically application specific. For example, medical and satellite images require different color spaces than unconstrained imagery, such as color photos, in order to retain uniformity, completeness and compactness in representing the color features.

We begin by defining a color point as a vector in the RGB color space as follows

Definition 1 Color point, $\mathbf{v}_c$. *A color point is a vector* $\mathbf{v}_c = (r, g, b)$ *in RGB color space.*

The transformation T_c and quantization Q_c of the RGB color space reorganizes and groups the vectors $\mathbf{v}_c$ to create a finite set of M colors.

2.6.1 Color transformation

The color transformation T_c is an operation on the vectors $\mathbf{v}_c \in RGB$ to produce the transformed vectors $\mathbf{w}_c$. The simplest color transformations are linear. For example, from RGB, the color spaces: YIQ (NTSC composite color TV standard), YUV (PAL and SECAM color television standards), $YCrCb$ (JPEG digital image standard and MPEG digital video standard) and opponent color space OPP are obtained by linear transformation [126]. Other color spaces, such as HSV, CIE 1976 ($L^*a^*b^*$) and CIE 1976 ($L^*u^*v^*$) are generated by non-linear transformations. The overall effect of color transformation T_c is to produce a new color space.

Since the RGB color space is continuous, or possibly discrete with a large number of values (i.e., digital photographs are typically represented in RGB color space discretized to 256 levels per axis, which gives over 1.67 million distinct color points), the color space must be quantized or partitioned into a smaller number of colors.

2.6.2 Color quantization

Since the color spaces are multi-dimensional, a partitioning of a color space is described by the vector quantization of the space. In general, a vector quantizer Q_c of dimension k and size M is a mapping from a vector in k-dimensional space into a finite set $\mathcal{C}$ containing M outputs [50]. Thus, a vector quantizer Q_c is defined as $Q_c : \Re^k \rightarrow \mathcal{C}$, where $\mathcal{C} = (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{M-1})$ and $\mathbf{y}_m \in \Re^k$ for each $m \in 0, 1, \ldots, M-1$. In general, the set $\mathcal{C}$ is called the codebook and has size M. In the case of vector quantization of the color space, $k = 3$, and each entry in the codebook y_m corresponds to a color point.

Therefore, the codebook $\mathcal{C}$ represents the gamut or collection of colors. In this way, a color, which corresponds to one quantized partition R_m of the color space and is an approximate perceptual unit. This is distinguished from the transformed color point $\mathbf{w}_c$, which is a mathematical vector in the continuous space.

Associated with the M point vector quantizer is a covering of $\Re^k$ into the M partitions, where each partition R_m contains all points $\mathbf{w}_c$ that are assigned the same codeword y_m:

$$R_m = \mathbf{w}_c \in \Re^k : Q_c(\mathbf{w}_c) = \mathbf{y}_m.$$

From the definition of the partitions it follows that they completely cover $\Re^k$ and are non-overlapping:

$$\bigcup_m R_m = \Re^k \qquad \text{and} \qquad R_m \bigcap R_n \qquad \forall m \neq n,$$

so that the partitions form a complete partitioning of $\Re^k$.

Definition 2 Color, *with index value m. A unit of color corresponds to a partition R_m and is given by the set of color points $\{\mathbf{v}_c\}_m$ that are assigned the same codeword y_m in the overall process of transformation and vector quantization: $y_m = Q_c(T_c\mathbf{v}_c)$.*

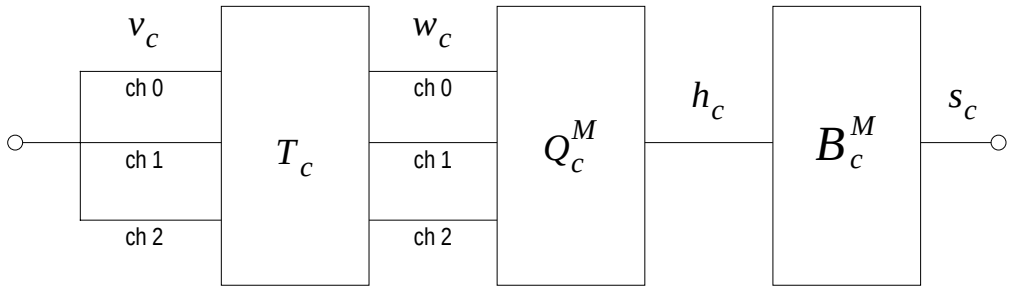


Figure 2-1: Transformation (T_c) and quantization (Q_c) produce the color feature space in which the color histograms $\mathbf{h}_c$ and binary color sets $\mathbf{s}_c$ are defined.

2.7. Color spaces

We now present and evaluate several transformations T_c of the RGB color space which generate new color spaces. Transformation of RGB is the starting point in designing the appropriate color spaces for defining the color histogram and binary color set representations.

2.7.1 RGB color space

The first candidate color space is the RGB color space. However, the RGB space is not perceptually uniform. As such, the proximity of colors in RGB color space does not indicate color similarity. Since RGB is complete, other complete color spaces are generated by transforming RGB using T_c, which is not necessarily a linear transform.

2.7.2 Linear transform color spaces

Several linear color transformations are in wide use for representing, transmitting and broadcasting color images and videos [68]. Many of these transforms are computed by simple matrix multiplication.

2.7.2.1 OPP color space

The opponent color space (OPP) is obtained by linear transformation of the RGB color space [4]. There is evidence that human color vision uses an opponent-color model by which the responses of the R, G and B cones are combined into two opponent color pathways [101]. The transformation from RGB to OPP is

obtained by $\mathbf{w}_c = T_c^{OPP}\mathbf{v}_c$, where

$$T_c^{OPP} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 2 \\ 1 & -2 & 1 \end{bmatrix}. \tag{2.2}$$

The advantage with the OPP color space is that it is obtained easily by T_c^{OPP}. The disadvantages are that it is not uniform or natural. The color distance in OPP color space does not provide a robust measure of color dissimilarity. One component of OPP is a luminance channel. The two chrominance channels do not correspond to hue and saturation, but rather to, blue *vs.* yellow and red *vs.* green.

2.7.2.2 Linear color transform standards

The YIQ, YUV, and $YCrCb$ linear color transforms have been adopted in recent color picture systems. Each of these produces a linear transform of RGB which generates one luminance channel and two chrominance channels. The transformations were designed specifically to the parameters of the expected display devices: YIQ – NTSC color television, YUV – PAL and SECAM color television, and $YCrCb$ – color computer display. None of these color spaces is uniform. As such, the color distances in these transform color spaces do not correspond to color dissimilarities.

YIQ color transform The YIQ color space was developed for the NTSC composite color TV standard [101]. The transformation from RGB to YIQ is obtained by $\mathbf{w}_c = T_c^{YIQ}\mathbf{v}_c$ [131], where

$$T_c^{YIQ} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix}. \tag{2.3}$$

2.7.2.3 YUV color transform

The YUV color space is used in the PAL and SECAM color television standards [101]. The transformation from RGB to YIQ is obtained by $\mathbf{w}_c = T_c^{YUV}\mathbf{v}_c$, where

$$T_c^{YUV} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix}. \tag{2.4}$$

2.7.2.4 $YCrCb$ color transform

The $YCrCb$ color space is used in the JPEG digital image standard. The transformation from RGB to $YCrCb$ is obtained by $\mathbf{w}_c = T_c^{YCrCb}\mathbf{v}_c$, where

$$T_c^{YCrCb} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ 0.5000 & -0.4187 & -0.0813 \\ -0.1687 & -0.3313 & 0.5000 \end{bmatrix}. \tag{2.5}$$

Although the linear color transforms are simplest, they do not generate natural and uniform color spaces. We look next, at the Munsell color order system, which was defined with the objectives of naturalness, compactness and completeness.

2.7.3 Munsell color order system

The Munsell color order system [99] was developed to provide a notational system for colors, which organizes the colors according to natural attributes. Munsell's *Book of Color* [100] contains $1,200$ samples of color chips, each assigned a value of hue, saturation and chroma. As such, by Definitions 1 and 2, each Munsell color chip corresponds to a color generated by some T_c and Q_c of the RGB color space. The chips are arranged such that unit steps between them are intended to be perceptually equal.

The advantage of the Munsell color order system is that it orders a finite set of colors by perceptual similarities over an intuitive three dimensional space. The disadvantages are that the color order system does not indicate the transformation T_c required from RGB color space or partitioning Q_c to produce the set

of color chips. Although one transformation, named the Mathematical Transform to Munsell (MTM), from RGB to Munsell HVC has been investigated for image data by Miyahara [98], there does not exist a simple mapping from color points in RGB color space to Munsell color chips. Munsell was designed to be compact (1200 perceptually distinct chips) and complete. However, the Munsell color order system does not satisfy the property of uniformity. The color order system does not provide for the assessment of the similarity of color chips that are not neighbors.

2.7.4 CIE color spaces

Towards the goal of deriving uniform color spaces, the CIE * in 1976 defined the CIE 1976 ($L^*u^*v^*$) and CIE 1976 ($L^*a^*b^*$) color spaces [129]. These are generated by transforming the RGB color space first by linear transformation to the XYZ color space followed by one of two non-linear transformations: to CIE 1976 ($L^*u^*v^*$) color space or to CIE 1976 ($L^*a^*b^*$) color space. The first linear transformation between RGB and XYZ is given by

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.813 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{2.6}$$

The nonlinear transformations from XYZ to CIE 1976 ($L^*a^*b^*$) and CIE 1976 ($L^*u^*v^*$), which are given in [175], are determined by relation to a nominally white object-color stimulus which gives the tristimulus values X_n, Y_n, Z_n. In this case, the lightness is given by

$$L^* = 116 \left(\frac{Y}{Y_n} \right)^{1/3} - 16. \tag{2.7}$$

The values for u, v and a, b are given as follows:

CIE 1976 ($L^*u^*v^*$):

$$\begin{aligned} u^* &= 13L^*(u' - u'_n) \\ v^* &= 13L^*(v' - v'_n), \end{aligned} \tag{2.8}$$

where

$$\begin{aligned} u' &= \tfrac{4X}{X+15Y+3Z} \quad u'_n = \tfrac{4X_n}{X_n+15Y_n+3Z_n} \\ v' &= \tfrac{9Y}{X+15Y+3Z} \quad u'_n = \tfrac{9Y_n}{X_n+15Y_n+3Z_n}. \end{aligned} \tag{2.9}$$

The color distance between two color stimuli is calculated from

$$\Delta E^*_{uv} = [(\Delta L^*)^2 + (\Delta u^*)^2 + (\Delta v^*)^2]^{1/2}. \tag{2.10}$$

CIE 1976 ($L^*a^*b^*$):

$$\begin{aligned} a^* &= 500[(\tfrac{X}{X_n})^{1/3} - (\tfrac{Y}{Y_n})^{1/3}] \\ b^* &= 200[(\tfrac{Y}{Y_n})^{1/3} - (\tfrac{Z}{Z_n})^{1/3}]. \end{aligned} \tag{2.11}$$

The color distance between two color stimuli is calculated from:

$$\Delta E^*_{ab} = [(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2]^{1/2}. \tag{2.12}$$

The CIE color spaces represent with equal emphasis, the three color variants that best perceptually characterize color: *hue*, *lightness* and *saturation*. However, the CIE color spaces are inconvenient due to the necessary non-linearity in forward and reverse transformations with RGB space.

2.7.5 HSV color space

Another form of the *hue*, *lightness* and *saturation* transform from RGB to HSV is given in [68]. The transform to HSV is non-linear, but it is easily invertible. The HSV color space is natural and approximately

*Commission Internationale de l'Eclairage

perceptually uniform. Therefore, we can define a quantization Q_c of HSV to produce a collection of colors that is also compact and complete.

2.7.5.1 HSV transformation

This transformation T_c from RGB to HSV is accomplished through the following equations [68]: let the color triple $\mathbf{v}_c = (r, g, b)$ be the to a color point in RGB space and let $\mathbf{w}_c = (h, s, v)$ be the to the transformed color point in HSV color space, such that $\mathbf{w}_c = T_c(\mathbf{v}_c)$, then T_c is given as follows:

Transformation 1 RGB to HSV *For $r, g, b \in [0 \ldots 1]$ then T_c gives $h, s, v \in [0 \ldots 1]$ as follows*

$$\mathbf{v} = \max(r, g, b), \qquad \mathbf{s} = \frac{\mathbf{v} - \min(r, b, g)}{\mathbf{v}}$$

$$\text{let } \acute{r} = \frac{\mathbf{v} - r}{\mathbf{v} - \min(r, b, g)}, \qquad \acute{g} = \frac{\mathbf{v} - g}{\mathbf{v} - \min(r, b, g)}, \qquad \acute{b} = \frac{\mathbf{v} - b}{\mathbf{v} - \min(r, b, g)}$$

$$6\mathbf{h} = \begin{cases} 5 + \acute{b} & \text{if } r = \max(r, g, b) \text{ and } g = \min(r, b, g) \\ 1 - \acute{g} & \text{if } r = \max(r, g, b) \text{ and } g \neq \min(r, b, g) \\ 1 + \acute{r} & \text{if } g = \max(r, g, b) \text{ and } b = \min(r, b, g) \\ 3 - \acute{b} & \text{if } g = \max(r, g, b) \text{ and } b \neq \min(r, b, g) \\ 3 + \acute{g} & \text{if } b = \max(r, g, b) \text{ and } r = \min(r, b, g) \\ 5 - \acute{r} & \text{otherwise} \end{cases}$$

Transformation 2 HSV to RGB *For $h, s, v \in [0 \ldots 1]$ then T_c^{-1} gives $r, g, b \in [0 \ldots 1]$ as follows*

$$\alpha = 6\mathbf{h} - \text{round}(6\mathbf{h}), \qquad \omega_1 = (1 - \mathbf{s}) * \mathbf{v},$$
$$\omega_2 = (1 - (\mathbf{s} * \alpha)) * \mathbf{v}, \qquad \omega_3 = (1 - (\mathbf{s} * (1 - \alpha))) * \mathbf{v}$$

$$\mathbf{r} = \begin{cases} \mathbf{v} & \text{if } \alpha = 0 \text{ or } \alpha = 5 \\ \omega_1 & \text{if } \alpha = 2 \text{ or } \alpha = 3 \\ \omega_2 & \text{if } \alpha = 1 \\ \omega_3 & \text{if } \alpha = 4 \end{cases} \quad \mathbf{g} = \begin{cases} \mathbf{v} & \text{if } \alpha = 1 \text{ or } \alpha = 2 \\ \omega_1 & \text{if } \alpha = 4 \text{ or } \alpha = 5 \\ \omega_2 & \text{if } \alpha = 3 \\ \omega_3 & \text{if } \alpha = 0 \end{cases} \quad \mathbf{b} = \begin{cases} \mathbf{v} & \text{if } \alpha = 3 \text{ or } \alpha = 4 \\ \omega_1 & \text{if } \alpha = 0 \text{ or } \alpha = 1 \\ \omega_2 & \text{if } \alpha = 5 \\ \omega_3 & \text{if } \alpha = 2 \end{cases}$$

2.7.5.2 HSV quantization

We design the quantization Q_c of the HSV color space to produce a compact set of (166) colors. The HSV color space is cylindrical, as illustrated in Figure 2-2. The long axis represents *value*: blackness to whiteness. Distance from the axis represents *saturation*: amount of color present. The angle around the axis is the *hue*: tint or tone. Since hue represents the most significant characteristic of the color, it requires the most fine quantization.

In the hue circle, the primaries red, green and blue are separated by 120 degrees. A circular quantization at 20 degree steps sufficiently separates the hues such that the three primaries and yellow, magenta and cyan are represented each with three sub-divisions. Saturation and value are each quantized to three levels yielding greater perceptual tolerance along these dimensions. The quantization Q_c provides $M = 166$ distinct colors in HSV color space [145].

2.7.5.3 HSV color similarity

Since the HSV color space is nearly perceptually uniform, the similarity of two HSV colors is determined by their proximity in the HSV color space. The similarity between two colors given by indices $m_i = (h_i, s_i, v_i)$ and $m_j = (h_j, s_j, v_j)$ is given by the euclidean distance between the color points in the cylindrical HSV color space as follows:

$$a_{i,j} = 1 - \frac{1}{\sqrt{5}}[(v_i - v_j)^2 + (s_i \cos h_i - s_j \cos h_j)^2 + (s_i \sin h_i - s_j \sin h_j)^2]^{\frac{1}{2}}. \tag{2.13}$$

The normalization by $1/\sqrt{5}$ gives the similarities: $a_{i,i} = 1$, and $a_{i,j} = 0$ for colors indexed by i and j that are separated by the maximum possible distance in HSV color space.

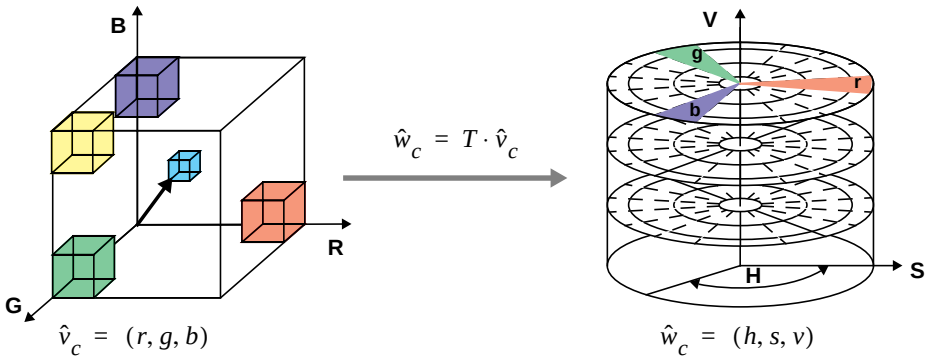


Figure 2-2: Transformation T_c from RGB to HSV and quantization gives 18 *hues*, 3 *saturations*, 3 *values* and 4 *grays* = 166 colors.

2.7.5.4 Comparisons of color transforms

Table 2.1 summarizes the color transforms. Only the OPP, YIQ, YUV, and $YCrCb$ color spaces are generated from linear transforms of the RGB color space. The transformation from RGB to HSV is non-linear but reversible, and it is easily computed using Transformation 1. Of the candidate color spaces, only the quantized HSV color space satisfies the properties of uniformity, compactness, completeness and naturalness. The Munsell and CIE color spaces are natural, however, the Munsell color order system is not uniform, while the CIE color spaces are not compact.

	RGB	OPP/YIQ/YUV/YCrCb	Munsell	CIE	166-color HSV
Uniformity	No	No	No	Yes	Yes
Compactness	No	No	Yes	Possible	Yes
Completeness	Yes	Yes	Yes	Yes	Yes
Naturalness	No	No	Yes	Yes	Yes
Transformation	N/A	L	NL	L + NL	NL

Table 2.1: Comparisons of color transforms (L = linear, NL = non-linear).

2.8. Representing color image features

Using the 166-color quantized HSV color space, we represent image features by measuring the distribution of the 166 colors in the image.

2.8.1 Color histograms

The distribution of colors in the image is represented by a histogram. Given a color image $\mathbf{I}[x, y]$ which consists of three color channels $\mathbf{I} = (I_R, I_G, I_B)$, the color histogram is given by, where X and Y are the width and height of the image, respectively,

$$h_c[m] = \sum_{x \ldots 0}^{X-1} \sum_{y \ldots 0}^{Y-1} \begin{cases} 1 & \text{if } Q_c(T_c \mathbf{I}[x, y]) = m \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

Definition 3 Color histogram, $\mathbf{h}_c$. *A color histogram determines the distribution of colors in an image, region or object.*

2.8.2 Color sets

Alternatively, the color content of images or regions is represented more compactly by binary color sets. The binary color sets are obtained by selecting from the M colors; the overall process is depicted in Figure 2-1. Let $\mathcal{B}^M$ be the M dimensional binary space such that each axis in $\mathcal{B}^M$ corresponds to one color, indexed by m.

Definition 4 Color set, $\mathbf{s}_c$. *A color set is a binary vector in $\mathcal{B}^M$ and determines a set of colors $\{m\}$ in an image, region or object.*

A binary set is equivalent to a thresholded histogram, where each bin is thresholded to two levels. For example, given threshold τ_m for color m, a color set is obtained from (see also Figure 2-3)

$$s_c[m] = \begin{cases} 1 & \text{if } h_c[m] \geq \tau_m \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

The color set indicates only those colors that are found above threshold levels. However, it works well to represent regional color. If a color is not well represented in a region – for example, if it is below threshold τ_m – it is ignored. In this way, using the binary color set representation, color content is represented by the set of only the most prominent colors in the region.

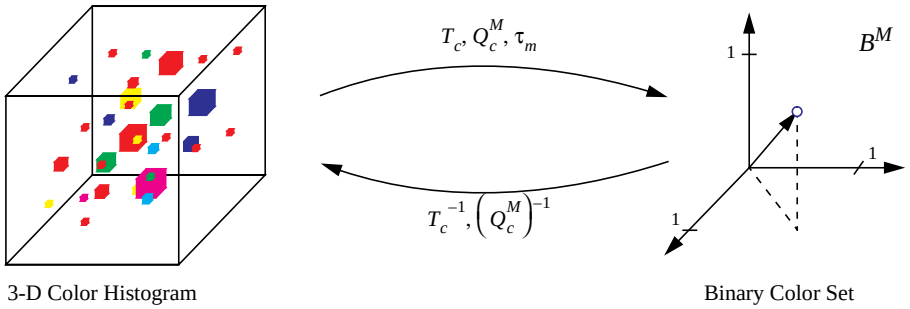


Figure 2-3: Relationship between 3-D RGB color histogram and binary color set.

2.8.3 Color histogram compaction

Since our objective is to represent the color information of regions and images, we examine the characteristics of the color histograms of typical images and regions. Figure 2-4 compares the energy compaction of the color histograms for manually extracted image regions to those for full images. We see that the color information for the regions within the images is defined by largely in a few most significant colors. For full images, the color information is spread out into many colors. This observation provides several directions:

1. to represent regional color, we use compact features sets, such as binary color sets. We investigate the retrieval effectiveness of the binary color sets in Chapter 4.

2. Since regions have few colors, we assume that the areas in images with few colors correspond to regions. We explore this assumption in the development of an automated system for extracting color regions from images in Chapter' 5.

Since we will further explore the versatility of the color histogram and binary color sets, we define also a fast mapping from histograms to the nearest binary sets (BSFM). We use this procedure in Chapter 4 in a technique for efficiently computing histogram queries using binary set bounding (BSB).

2.8.4 Binary set fast map (BSFM)

We define an algorithm for determining the nearest binary set to an arbitrary histogram: the binary set fast map (BSFM). The objective is to find the one binary set $\mathbf{s}_c$ that minimizes the distance to the histogram $\mathbf{h}_c$, that is, such that $\epsilon_c(\mathbf{h}_c, \mathbf{s}_c)$ is minimized. We define ϵ_c to be the mean square error metric as follows:

$$\epsilon_c(\mathbf{h}_c, \mathbf{s}_c) = \sqrt{\sum_{m=0}^{M-1} (h_c[m] - \frac{s_c[m]}{|\mathbf{s}_c|})^2},$$

where $|\mathbf{s}_c|$ denotes the number of non-zero elements in the binary set $\mathbf{s}_c$. The algorithm proceeds as follows:

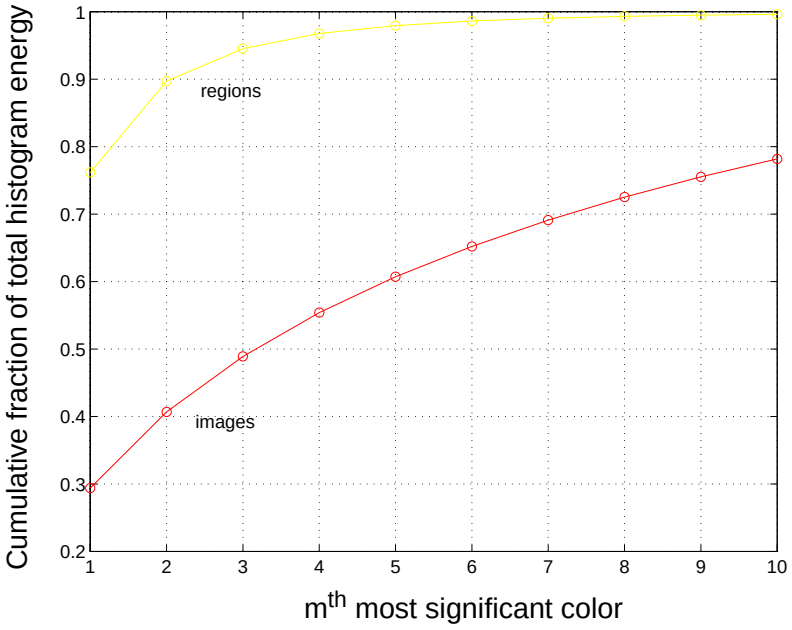


Figure 2-4: Compaction of color histogram energy for regions vs. images.

1. the elements in histogram $\mathbf{h}_c$ are sorted in order of largest value.
2. Starting from the single largest element we generate a binary set of cardinality $|\mathbf{s}_c| = 1$ that assigns a value of one to this element, and we compute the error between the histogram and the binary set.
3. The process is then repeated for the first two largest elements in the histogram, then the first three largest elements, and so forth, until the collection of nearest binary sets of sizes $|\mathbf{s}_c| = 0 \ldots M-1$ is generated.
4. From this collection, the binary set that minimizes the error to the histogram is the nearest binary set.

The procedure is given in detail as follows: let $\mathcal{M}_s$ be the mapping that sorts $h_c[m]$ such that $\mathbf{g}_c = \mathcal{M}_s \mathbf{h}_c$ and $g_c[0] \geq g_c[1] \geq \ldots \geq g_c[M-1]$. We also note that $\mathbf{h}_c$ can also be derived from $\mathbf{g}_c$ by inverting the sorting process: $\mathbf{h}_c = \mathcal{M}_s^{-1} \mathbf{g}_c$.

Algorithm 1 Fast map to nearest size n binary set
Let $\mathbf{r}_c^n$ be a binary set of cardinality $|\mathbf{r}_c^n| = n$ such that

$$r_c^n[m] = \begin{cases} 1 & m < n \\ 0 & otherwise \end{cases}$$

Then, let

$$\epsilon_c^n(\mathbf{g}_c, \mathbf{r}_c^n) = \sqrt{\sum_{m=0}^{M-1} (g_c[m] - \frac{r_c^n[m]}{n})^2}.$$

The nearest size $|\mathbf{s}_c| = n$ binary set $\mathbf{s}_c$ to histogram $\mathbf{h}_c$ is given by $\mathbf{s}_c = \mathcal{M}_s^{-1} \mathbf{r}_c^n$, where ϵ_c^n also gives the distance between $\mathbf{s}_c$ and histogram $\mathbf{h}_c$, that is $\epsilon_c^n(\mathbf{g}_c, \mathbf{r}_c^n) = \epsilon_c(\mathbf{h}_c, \mathbf{s}_c)$.

The nearest $|\mathbf{r}_c^n| = n$ binary set to sorted histogram $\mathbf{g}_c$ is given by $\mathbf{r}_c^n$. By using the inverse mapping $\mathcal{M}_s^{-1}$ that was used to sort $\mathbf{h}_c$, the nearest size $|\mathbf{s}_c| = n$ binary set $\mathbf{s}_c$ to histogram $\mathbf{h}_c$ is given by $\mathbf{s}_c = \mathcal{M}_s^{-1} \mathbf{r}_c^n$.

Algorithm 2 Fast map to nearest binary set *The overall nearest binary set $\mathbf{s}_c$ to histogram $\mathbf{h}_c$ is found by determining the nearest size $|\mathbf{s}_c^n| = n$ binary sets $\mathbf{s}_c^n$ for $n = 1 \ldots M$. The one that provides the minimum ϵ_c^n gives the nearest binary set.*

Overall, the algorithm requires $M \log M$ operations to sort the histogram $\mathbf{h}_c$, M computations of mean square error and M operations to determine the nearest binary set.

2.9. Summary

In this chapter, we evaluated several transformations (T_c) and quantizations (Q_c) of color. The objectives were to obtain a color feature space in which color histograms are defined. We demonstrated that the HSV color space, quantized to 166 values provides a suitable color space because it is uniform, complete, compact and natural.

We defined the color histograms ($\mathbf{h}_c$) as distributions of the HSV colors that appear in an image or region. We also presented the binary color set representation. Binary color sets ($\mathbf{s}_c$) represent only a selection of colors and provide a compact measure of image and/or region color. We also defined a fast algorithm (BSFM) for computing the nearest binary color set to a color histogram.

Using the content-based image query paradigm, we explore the image retrieval effectiveness of the color histogram and color set representations in Chapter 4. Then, in Chapter 5, we present and evaluate techniques for extracting color regions from images using the color histograms and binary color sets. These feature extraction and feature query processes are integrated to produce the spatial and color feature query system described in Chapter 7. However, in the next chapter, we first develop the texture histogram and binary texture set representations of texture that are symmetric with those for color. This allows us to uses generic methods of feature and region extraction, and feature indexing and querying that are compatible with representations of both visual dimensions.

Chapter 3

Representing Texture

3.1. Introduction

The texture representation is symmetric with color in that texture is also defined from multiple channels. The texture images are first decomposed into nine spatial-frequency subbands using a wavelet filter bank (FB). Then, from the nine subbands, a texture channel generator (TCG) produces the nine channels. As investigated in Chapter 2 for color, we investigate procedures for transforming (T_t) and quantizing (Q_t) the texture channels to produce a texture element feature space in which to define suitable histogram ($\mathbf{h}_t$) and binary texture set ($\mathbf{s}_t$) representations of texture.

In this chapter, we present the overall process of representing texture using the texture histograms. We present the wavelet texture energy feature set and compare it to Gabor filter-based representations. We also present experimental evaluations of the wavelet texture feature sets that demonstrate excellent performance in classifying the standard evaluation-set of Brodatz textures. We then investigate several transformations that allow for approximate rotation or scale invariance. Finally, we present the process of generating the texture histogram and binary texture set-based feature spaces that are symmetric to those for color histograms discussed in Chapter 2.

The objectives of obtaining these representations of texture are to provide systems for measuring the similarity of textures and extracting texture regions from images. In Chapter 4, we present and evaluate the various techniques for computing texture similarity using the representations presented here. In Chapter 5, we present the system for extracting texture regions from images using texture histograms.

3.2. What is texture?

Textures are homogeneous patterns or spatial arrangements of pixels that regional intensity or color alone does not sufficiently describe. As such, textures have statistical properties, structural properties, or both [60]. They may consist of the structured and/or random placement of elements, but also may be without fundamental subunits. For example, Figure 3-1 illustrates the wide variety of textures from the Brodatz collection. Moreover, due to the diversity of textures appearing in natural images it is difficult to narrowly define texture.

Texture is an important element to human vision. Texture has been found to provide cues to scene depth and surface orientation [4]. People also tend to relate texture elements of varying size to a plausible 3-D surface. Even in graphics systems greater realism is achieved when textures are mapped to the 3-D surfaces. Texture features have been used to identify contents of ariel imagery such as bodies of water, crop fields and mountains [92, 4, 87, 92]. Texture describes the content of many real-world images: for example, clouds, trees, bricks, hair, fabric all have textural characteristics.

It has been hard to adequately model texture. The substantial body of work on texture has not yet produced any clear solutions for the problems of texture analysis, classification, segmentation and synthesis [164]. Furthermore, the analysis and extraction of texture in unconstrained imagery, which is the most difficult problem, has not yet been sufficiently addressed.

Most research on texture is conducted on the Brodatz [9] texture collection (sixteen Brodatz textures are illustrated in Figure 3-1), which provides a set of mostly homogeneous texture images. In the Brodatz collection there is little visible distortion in the images that results from viewing perspective and lighting.

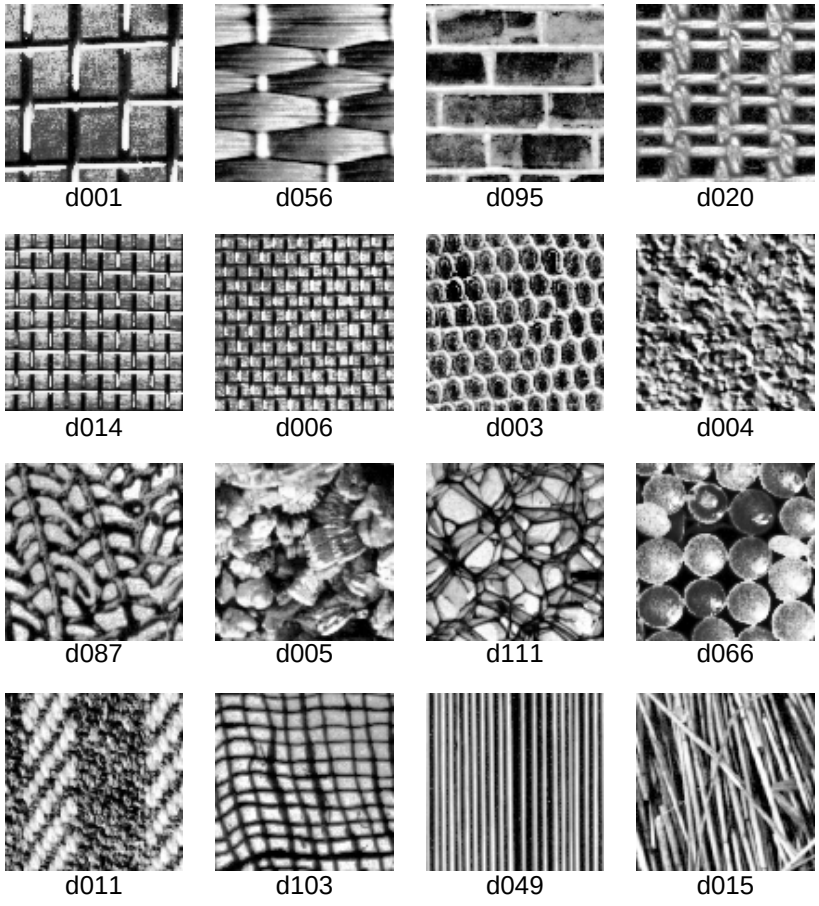


Figure 3-1: Example textures from the Brodatz texture collection: d001: Woven aluminum wire, d056: Straw matting, d095: Brick wall, d020: French canvas, d014: Woven aluminum wire, d006:Woven aluminum wire, d003: Reptile skin, d004: Pressed cork, d087: Sea fan, fossilized with a coral covering, d005: Expanded mica, d111: Plastic bubbles, d066: Plastic pellets, d011: Homespun woolen cloth, d103: Loose burlap, d049: Straw screening, d015: Straw.

In the literature, most techniques for texture analysis are investigated and/or evaluated using Brodatz textures [160, 116, 89, 118, 30, 125, 61, 10, 92, 23] and are not generally applicable to "unconstrained" images.

The appearance of texture in unconstrained imagery is substantially different from that represented in the Brodatz set. Textures in these real-world images are distorted by artifacts such as those resulting from non-uniform lighting, shading and warping in the 3-D space. This makes the problem of texture region extraction even more difficult.

We note that recent models of surface reflectance [4, 91] may prove useful for compensating for lighting and shading artifacts. Furthermore, texture has proven useful for decoding shapes and perspective projections [4]. Similar techniques may aid in de-warping the texture patterns in images such that orthogonal views (where $\phi = 90°$ in Figure 3-2) are obtained. Although texture region extraction is one of the objectives of this work, we do not focus on these aspects of texture analysis.

For the purpose of this work, we assume that the textures are free from perspective transformation. For example, as illustrated in Figure 3-2, for one, we assume that the view angle for the textures is perpendicular to the texture surface ($\phi = 90°$). However, the rotation (θ) and scale (ζ) are not restricted, and we explore texture representations that approximate rotation- or scale-invariance.

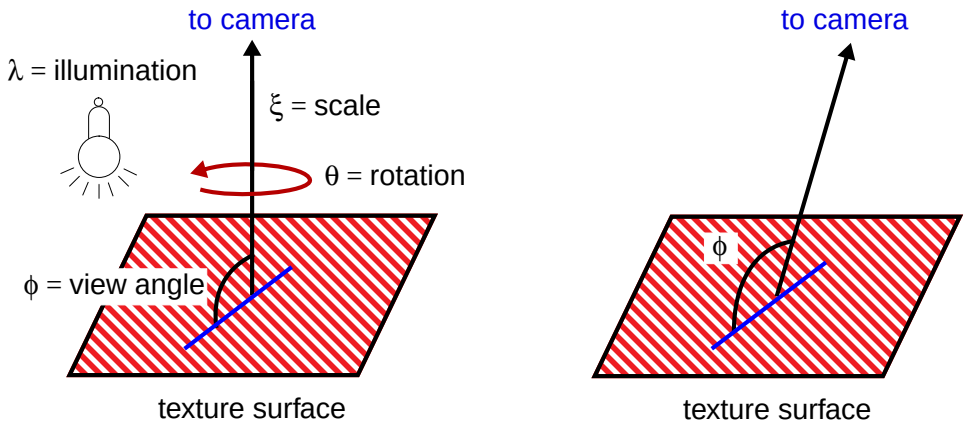


Figure 3-2: Image formation from texture surface.

3.3. Related work on texture

Recent attempts at modeling texture include random field modeling [30, 89, 119, 46], fractal geometry [24], spatial gray-level dependencies and co-occurrence matrices [52, 60] and spatial-frequency techniques [74, 84, 139, 23], which include, in particular, Gabor filtering [8, 121, 127].

A comparison of these four classes of texture features was made by Ohanian and Dubes in [106] using small test collections of natural and artificial images. The authors found that co-occurrence matrices performed best in the experiments, but they acknowledged that performance could be largely influenced by optimization within each texture representation-type. Kundu and Chen report a spatial-frequency technique based upon the QMF wavelet transform that improves texture image classification performance over spatial gray-level co-occurrence matrices.

3.4. Related work on texture image retrieval

Several recent content-based image retrieval systems utilize texture feature sets to aid in the retrieval of images. Within the applications of satellite image retrieval, recent systems by Li and Turek [87] and Ma and Manjunath [92] use texture to retrieve images based upon the detection of various features of the earth's terrain.

The IBM QBIC system [103] uses several texture features for the retrieval of photographic images. The QBIC texture features are based upon some of those proposed in [160], namely, coarseness, directionality and

contrast. Other texture features in [160] such as linelikeness, regularity and roughness are not used. Both the whole images and manually identified regions are indexed by texture in the QBIC system.

The Virage system [3] also uses texture measures for photographic image retrieval. In the Virage system, the texture features describe the whole image, rather than regions of texture within the images.

A unified approach for the characterization of global color, texture and shape was proposed by Caelli and Reye in [12]. Their objective was to classify objects which have color, texture and shape attributes. In their experiments, they used hand-written text characters rendered with different color and texture patterns. Their system was not applied to retrieval of unconstrained images.

Liu and Picard [89, 119] and Francos, Meiri and Porat [46] have investigated the Wold random-field model for modeling texture. The Wold model separates the image into components that correspond, approximately, to periodicity, directionality and randomness [119]. Liu and Picard have applied Wold feature to segmentation and retrieval of natural scene images [89].

Gorkani and Picard have investigated a texture measure based upon "dominant perceived orientation" for sorting photos [51]. The orientation is extracted over levels of a multiresolution steerable pyramid. They found that this texture feature is successful in classifying scenes as "city" *vs.* "suburb."

We propose a new representation of texture which is derived from spatial-frequency energy of the texture. From the energy values, a texture histogram is generated. The texture histogram is used to discriminate textures and to retrieve images. This approach is compatible with color histogram-based methods for content-based image systems. Furthermore, since the texture histograms are derived from the QMF wavelet transform, the features are more directly accessible from wavelet-compressed image representations.

3.5. Representations of texture

As described earlier, the objectives in developing the texture feature set are to provide a basis for (1) assessing the similarities of the texture features in images, (2) storing and indexing texture information and (3) extracting texture regions from images. In order to develop the texture feature set, we formulate the representation of texture from a set of channels. This provides an approach that is symmetric with that for the color feature sets.

The most suitable models are based upon the spatial-frequency (*s-f*) techniques. For one, the *s-f* texture models are also most aligned with the vision process. There is support that mechanisms of early human vision use receptive field units tuned to orientations and *s-f's* [54, 74]. In particular, models of the human vision system that use Gabor filters to model the receptive fields sufficiently account for psychophysical data obtained in texture discrimination experiments.

3.5.1 Space/spatial-frequency expansion

A space/spatial-frequency (*s/s-f*) expansion of the image captures its localized spatial-frequency content. In general, image *s-f* transforms attain a joint resolution in space and *s-f* that is bounded by the uncertainty principle [36, 171]. By spacing the filters at octave-band distances, the wavelet filter bank (FB) provides dyadic trade-offs in spatial and *s-f* resolution. Obtaining a high joint resolution in space and *s-f* is critical in texture region extraction where the localization of the texture pattern is important.

Gabor filters are one *s-f* technique which has been used to characterize texture by dominant *s-f*s [8]. Gabor functions measure the localized *s-f* information at specific *s-f* scales and orientations. One objective in modeling texture by Gabor functions is to characterize a texture by a single peak scale and orientation set [8].

Alternatively, Chang and Kuo [23] have used the tree-structured wavelet transform (TSWT) to classify texture, not by a single narrow subband, but by the overall best wavelet packet basis adapted to each texture. Feature sets based upon *s-f*s that use multiple subbands have also been investigated in [74, 84, 106]. We now look more closely at the Gabor FB and examine their relationship to QMF wavelet FBs.

3.5.2 Gabor Functions

Gabor filters achieve the theoretical lower bound of the uncertainty principle. They attain maximum joint resolution in space and *s-f* bounded by the relations $\Delta x \cdot \Delta u \geq \frac{1}{4\pi}$ and $\Delta y \cdot \Delta v \geq \frac{1}{4\pi}$ where $(\Delta x, \Delta y)$ gives resolution in space and $(\Delta u, \Delta v)$ gives resolution in *s-f*. This is highly significant in the process of texture extraction in which the conflicting objectives of accuracy in texture representation and texture spatial

localization are important. The Gabor filter based *s-f* analysis of texture was explored in [8, 120, 128, 41, 10].

In addition to good performance in texture discrimination and segmentation, the justification for Gabor filters is also supported through psychophysical experiments. It has been demonstrated that human texture segregation results from information corresponding to outputs of *s-f* channels [5]. Texture analyzers implemented using 2-D Gabor functions have produced a strong correlation between outputs of the 2-D Gabor FBs with actual human segmentation [128]. Furthermore, the receptive visual field profiles are adequately modeled by 2-D Gabor filters [36].

The 2-D Gabor function is a harmonic oscillator, composed of a sinusoidal plane wave of a particular frequency and orientation, within a Gaussian envelope. The frequency (ω), bandwidth (σ), location (x_0, y_0), and orientation (θ) are determined by parameters of the Gabor function. The Gabor function G is defined as follows:

$$G(x, y, \sigma, \omega, \theta, \phi) = J(x, y, \omega, \theta, \phi) \exp\left(\frac{(x - x_0)^2 + (y - y_0)^2}{-2\sigma^2}\right), \tag{3.1}$$

where the harmonic oscillator $J(x, y, \omega, \theta, \phi)$ is given by

$$J(x, y, \omega, \theta, \phi) = \sin\left(\omega[x \cos\theta - y \sin\theta] + \phi\right). \tag{3.2}$$

Here x_0 and y_0 specify the center of the Gaussian and σ specifies the standard deviation along both axes. The frequency of the sinusoidal plane wave is determined by ω, and θ is the angle of orientation, and ϕ is the phase of the plane wave. The Gabor filters are frequency and orientation selective as demonstrated from Fourier domain analysis. When the phase $\phi = 0$, the Fourier transform of the Gabor function is given by

$$F(u, v) = A\left(exp\left(-\frac{1}{2}\left[\frac{(u - u_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}\right]\right) + exp\left(-\frac{1}{2}\left[\frac{(u + u_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}\right]\right)\right), \tag{3.3}$$

where $\sigma_u = 1/(2\pi\sigma_x)$, $\sigma_v = 1/(2\pi\sigma_y)$ and $A = 2\pi\sigma_x\sigma y$. This function gives two lobes in the spatial-frequency domain, one centered at u_0 and another at $-u_0$ [164]. For example, Figure 3-3 illustrates the spatial-frequency response of a bank of Gabor filters to sinusoidal inputs (only u_0 term is depicted).

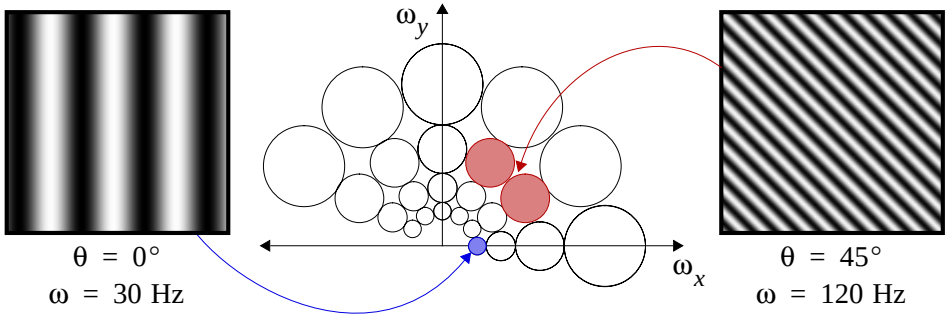


Figure 3-3: Gabor filter spectrum where contours indicate the half-peak magnitudes of filters responses. The Gabor filter responses to pure sinusoids: (a) $\theta = 0$ and $\omega = 30$ and (c) $\theta = 45$ and $\omega = 60$, are depicted in (b).

Gabor functions are used for texture segmentation by tuning the filters to the image's dominant spectral information. This entails selection of parameters for frequency, orientation and bandwidth for each filter. The selection is highly dependent on the image. In [8] a simple peak-finding algorithm applied to the power spectrum was used to guide the choice of filter frequency. Then, by using the chosen set of Gabor functions, the image is filtered. Each point in the image is labeled according to which filter output gives highest energy value for the point. The result is that the image is segmented into regions according dominant spectral information.

There are drawbacks with using the Gabor filters in practical application. The first concerns the consequence that the selection of filters is image dependent. This process is nontrivial [8]. Otherwise, the accurate

implementation of a complete Gabor expansion would entail a generally impractical number of filters. Since the Gabor functions are not orthogonal, there is a trade-off between redundancy and completeness in the design of the Gabor FBs.

Secondly, the application of the filters to images is not simple. The computation of the filter coefficients is a complex process because the Gabor functions are not orthogonal. Furthermore, discrete versions of the Gabor function must be obtained in order to be applied to images.

3.5.3 Wavelet expansion

A more practical way to exercise the trade-off between spatial and *s-f* resolution, without using Gabor functions, is through the use of a quadrature mirror filter (QMF) wavelet FB. The QMF wavelet FB produces octave bandwidth segmentations in *s-f*. It allows simultaneously for high spatial resolution at high *s-f*'s and high *s-f* resolution at low *s-f*'s. Furthermore, the benefit of using the wavelet tiling for image analysis is supported by evidence that visual *s-f* receptors are spaced at octave distances [35].

The QMF approach was investigated by Kundu and Chen [84]. The authors identified several aspects of the QMF FB as being relevant to texture extraction:

1. ability to achieve perfect reconstruction,
2. outputs that are localized filters and
3. the decimation of filter outputs reduces complexity.

In the classification of Brodatz textures by Kundu and Chen, the QMF features performed better than those proposed by Haralick [60].

3.5.4 QMF filter bank (FB)

The 2-D QMF FB partitions the 2-D *s-f* plane such that the image are reconstructed by the partitions (subbands), see Figure 3-4. Using the QMF filters, the reconstruction is nearly perfect [75]. The 2-D QMF wavelet FB produces the particular set of subbands that are spaced at octaves in the 2-D *s-f* plane.

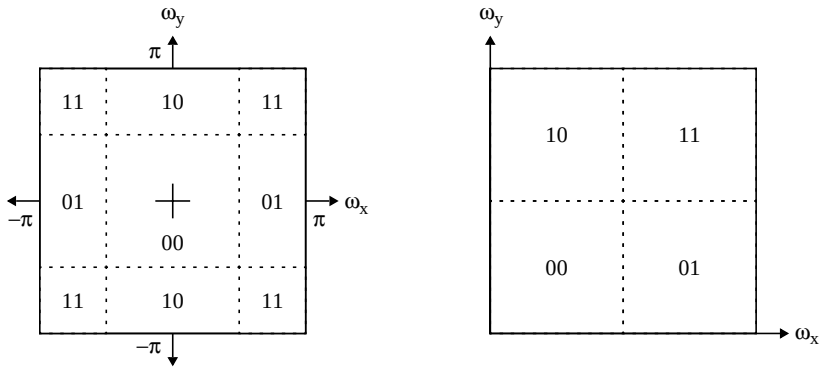


Figure 3-4: 2-D QMF filter bank (FB), (a) *s-f* plane partitioning, (b) subband interpretation.

Similar to the case for Gabor filters depicted in Figure 3-3, the 2-D sinusoidal signals produce responses in the QMF wavelet FB, as illustrated in Figure 3-5. We see that the wavelet FB differs from the Gabor FB in that the granularity in *s-f* resolution is fixed at octaves in *s-f*. In orientation, the wavelet transform separates the texture only into only vertical, horizontal and diagonal subbands.

In a 2-D QMF FB, four filters are required, H_{00}, H_{01}, H_{10} and H_{11} that have mirror-image conjugate symmetry about their mutual boundaries [174]. For real filters, this is equivalent to

$$\begin{aligned} H_{01}(z_x, z_y) &= H_{00}(z_x, -z_y) \\ H_{10}(z_x, z_y) &= H_{00}(-z_x, z_y) \\ H_{11}(z_x, z_y) &= H_{00}(-z_x, -z_y). \end{aligned} \tag{3.4}$$

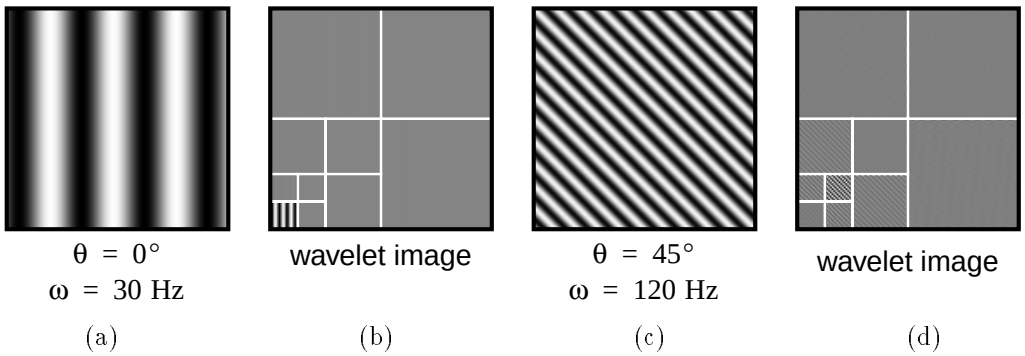


Figure 3-5: QMF wavelet FB responses to pure sinusoids: (a) $\theta = 0$ and $\omega = 30$, (b) wavelet image for (a), and (c) $\theta = 45$ and $\omega = 60$, (d) wavelet image for (c).

The 2-D baseband filter H_{00} may be designed to be separable, in which case it is generated from a product of two identical 1-D QMF filters [166],

$$h_{00}(m,n) = h_0(m)h_0(n). \tag{3.5}$$

In this case, the four QMF filters H_{ij} are constructed from

$$H_{ij}(z_x, z_y) = H_i(z_x)H_j(z_y), \quad \text{where} \quad j \in \{0,1\}. \tag{3.6}$$

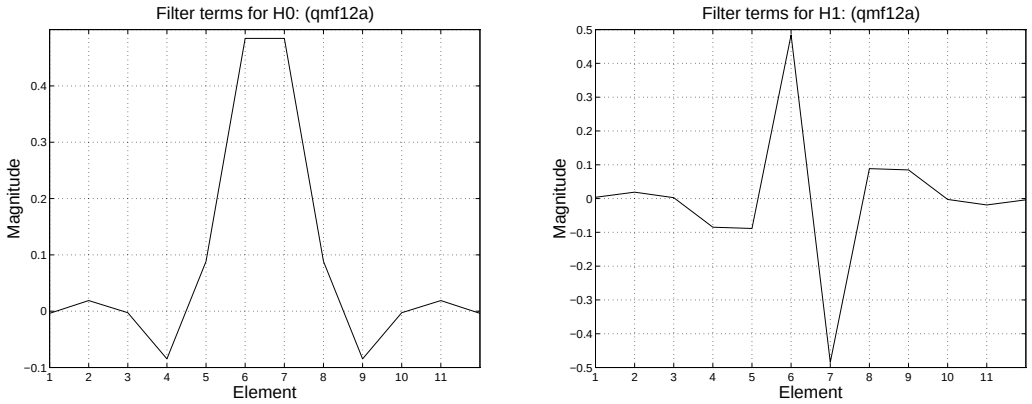


Figure 3-6: QMF filters (a) low pass filter h_0, (b) high-pass filter h_1.

The QMF FB is designed to have no aliasing or phase distortion. By using separable filters, the QMF FB is generated from the single filter kernel $H_0(z)$ by Eq. 3.4 and Eq. 3.6. The separable QMF FB approximately satisfies the following requirement in order to reduce or eliminate amplitude distortion [26], see Figure 3-6.

$$|H_0^2(z)| + |H_0^2(-z)| = 1. \tag{3.7}$$

In Figure 3-6(a) and (b), the filters are given for a twelve tap QMF (QMF12a from [75]): Figure 3-6(a) depicts the low-pass filter h_0 and Figure 3-6(b) depicts the high-pass filter h_1. From the magnitude responses in Figure 3-7 we can see that Eq. 3.7 is approximately satisfied; there is only slight ripple in $|H_0^2(z)|+|H_0^2(-z)|$.

3.5.5 Tree structured separable QMF FB

By iterating the four-channel 2-D QMF FB on the output of filter H_{00} we obtain a 2-D wavelet FB. The 2-D QMF wavelet FB is depicted in Figure 3-8. The output of each wavelet filter channel is produced from a

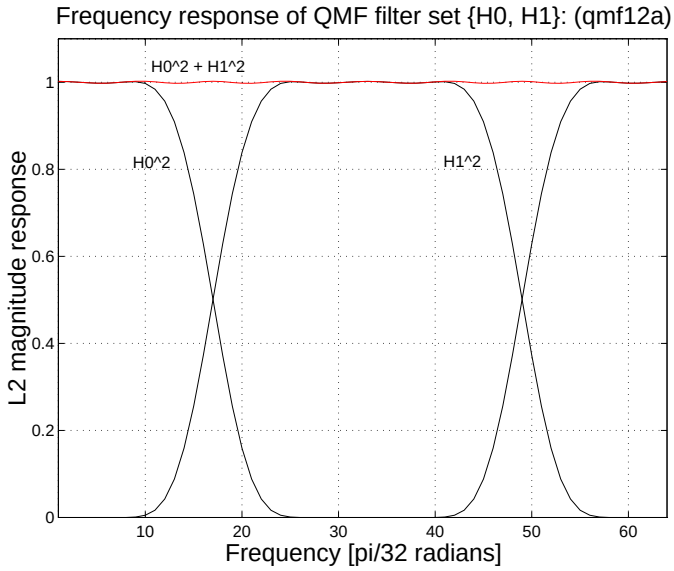


Figure 3-7: Magnitude of the frequency responses H_0 and H_1.

cascade of filters and downsamplers. For example, the four subbands at depth = 3 in the cascade are given by

$$\begin{aligned}\mathbf{Y}_{00i,00j}(z_x, z_y) &= \frac{1}{2}[\mathbf{H}^{(3)}_{00i,00j}(z_x^{1/2}, z_y^{1/2})\mathbf{X}(z_x^{1/2}, z_y^{1/2}) + \\ &\quad \mathbf{H}^{(3)}_{00i,00j}(-z_x^{1/2}, -z_y^{1/2})\mathbf{X}(-z_x^{1/2}, -z_y^{1/2})], \end{aligned} \tag{3.8}$$

where $i, j = \{0, 1\}$ and

$$\mathbf{H}^{(3)}_{00i,00j}(z_x, z_y) = \mathbf{H}_{0,0}(z_x, z_y)\, \mathbf{H}_{0,0}(z_x^2, z_y^2)\, \mathbf{H}_{i,j}(z_x^4, z_y^4). \tag{3.9}$$

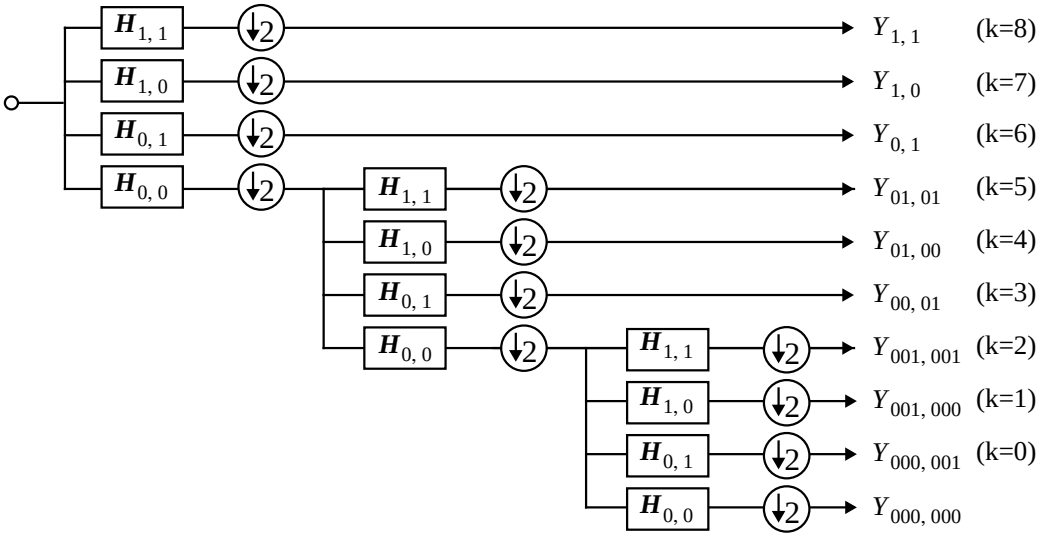


Figure 3-8: 2-D wavelet QMF FB.

3.5.6 Representing texture by *s/s-f* energies

The texture features are produced by measuring the energy within the subbands as follows, where the subbands are numbered as in Figure 3-8,

$$\mathbf{S}_k = \frac{1}{MN} \sum_{m}^{M-1} \sum_{n}^{N-1} |\mathbf{y}_k[m,n]|^2, \tag{3.10}$$

and M and N are the width and height of subband k. By Parseval's relation, $\mathbf{S}_k = \frac{1}{MN}||\mathbf{Y}_k||^2$. The QMF wavelet texture energy feature set $\mathbf{f}_t$ is constructed from the $\mathbf{S}_k$'s as follows

$$\mathbf{f}_t = (\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_k). \tag{3.11}$$

Figure 3-9 illustrates the wavelet transform of a composite of five Brodatz textures. We see that the textures produce different responses in the subbands. The feature set $\mathbf{f}_t$ captures the energy of the textures in each of the subbands.

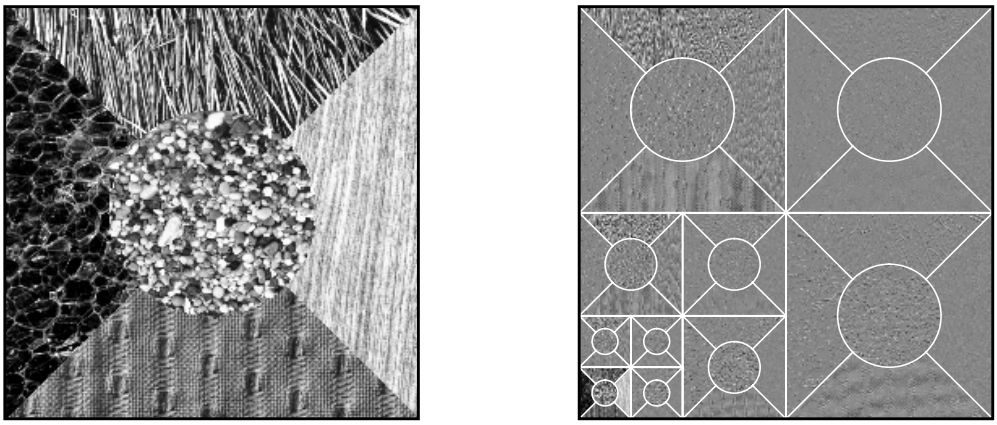


Figure 3-9: Wavelet transformation of a composite image of Brodatz textures.

We demonstrate two important invariance properties of the wavelet texture energy feature set: gray-level shift invariance and size invariance. We note that rotation and scale invariance is not inherently provided. We develop transformations T_t in Section 3.7.3 that generate feature sets that are approximately rotation- or shift-invariant.

3.5.6.1 Gray-level shift invariance

The texture energy feature set is gray-level shift invariant. The gray-level shift transform is defined as

$$I(n) \rightarrow I(n) + \beta. \tag{3.12}$$

In the FB, only h_0 has non-zero mean, i.e., $\sum h_0(n) \neq 0$, and the rest of the filters have zero mean. Since, $\sum_m h_i(m) = 0$ for $i \neq 0$, we have gray-level shift invariance as follows

$$\begin{aligned} y_i(n) &= h_i(n) * (I(n) + \beta) \\ &= \textstyle\sum_m h_i(m) I(n-m) + \beta \sum_m h_i(m) \\ &= \textstyle\sum_m h_i(m) I(n-m) = h_i(n) * I(n). \end{aligned} \tag{3.13}$$

3.5.6.2 Size invariance

The texture energy feature set is size invariant since the energy measure is normalized by the size of the subband. The elements of the feature set $\mathbf{S}_k$ are normalized by M and N, the width and height of the subband, respectively, in Eq 3.10.

3.6. Texture classification experiments

We compare texture energy feature sets that are derived from several image transforms besides the QMF wavelet, including DCT, QMF uniform subband. In the experiments reported here, 112 Brodatz texture images (512x512) are cut randomly into rectangular pieces (average size 132x132). The texture energy feature sets are used to classify the cuts to the correct Brodatz texture classes. We use a classification procedure based upon Fisher discriminant analysis.

Classification rates of over 90% are obtained for both wavelet subband and uniform subband decompositions. DCT/Mandala-block gives just over 80% classification rate, while simple spatial block features produces only 34% correct classification.

3.6.1 K-Class discrimination

Fisher discriminant analysis (FDA) derives the optimal linear classification function from the observation data that describes the set of classes. The FDA constructs linear composites from training features that provide for maximum average separation among classes. Although the resultant composites are not orthogonal, they are uncorrelated with each other [39].

FDA is contrasted with principal component analysis (PCA), which is performed on the data covariance matrix. PCA determines the most significant functions to best represent, not to classify, the data. Since our goal is classification, we use FDA, which determines the eigenvectors of scatter matrices which describe class separability. The criteria of class separability is formulated using the following scatter matrices: *within-class* ($\mathbf{W}$), *between-class* ($\mathbf{B}$), and *total* ($\mathbf{T}$) [48].

The FDA procedure is carried out as follows: let K = the number of classes. Each class k, where $k \in \{0, \ldots, K-1\}$, has a set of n_k observations and provides the length m feature vector as the j^{th} observation $x_{jk} = (x_{jk}^0, x_{jk}^1, \ldots, x_{jk}^{m-1})$, where $j \in \{0, 1, ..., n_k - 1\}$. Let $\overline{x}$ be the total mean vector over all observations and all classes

$$\overline{x} = \sum_{k=0}^{K-1} \sum_{j=0}^{n_k} x_{jk}. \tag{3.14}$$

Then, define t_k as follows

$$t_k = \sum_{j=0}^{n_k-1} (x_{jk} - \overline{x})^t (x_{jk} - \overline{x}). \tag{3.15}$$

Then, summing the t_k's gives, $\mathbf{T} = \sum_{k=0}^{K-1} t_k$, which is the *total-mean-corrected* sum-of-squares of cross-products of observation data.

Let $\overline{x}_k$ be the mean vector over all observations of class k

$$\overline{x}_k = \sum_{j=0}^{n_k} x_{jk}. \tag{3.16}$$

Then, define w_k as follows

$$w_k = \sum_{j=0}^{n_k-1} (x_{jk} - \overline{x}_k)^t (x_{jk} - \overline{x}_k). \tag{3.17}$$

Then, summing the w_k's gives, $\mathbf{W} = \sum_{k=0}^{K-1} w_k$, which is the *within-class* sum-of-squares matrix.

Then, $\mathbf{B} = \mathbf{T} - \mathbf{W}$ gives the *between-class* sum-of-squares and cross-products matrix. The product $\mathbf{W}^{-1}\mathbf{B}$ represents the ratio of *between-class* to *within-class* sum-of-squares for K classes. This matrix, which is non-symmetric with rank equal to the minimum of m (the feature vector length) and $K-1$ (the number of classes minus one) is used in the eigenvector decomposition as follows: denoting the eigenvalues of $\mathbf{W}^{-1}\mathbf{B}$ as λ_k and the eigenvectors as e_k, we determine which discriminant functions capture the most variation that separates the classes by expressing each eigenvalue as a percentage of total variance accounted for using

$$\lambda_k' = \frac{\lambda_k}{\sum_j \lambda_j}. \tag{3.18}$$

Then, classification follows by using a subset $\{e_k|\lambda'_k \geq \tau\}$ of the eigenvectors corresponding to the eigenvalues accounting for the largest total variation. A minimum distance rule is used to assign new observation data, unclassified texture images, to one of the classes. First, normalizing the eigenvectors, such that,

$$\mathbf{e}_i^t \frac{\mathbf{W}}{\sum_{j=0}^{K-1} n_k - K} \mathbf{e}_i = \begin{cases} 1 & \text{if } i = k \leq s \\ 0 & \text{otherwise,} \end{cases} \tag{3.19}$$

we get $\mathbf{Y}_k = e_k^t \mathbf{X}$, the k^{th} discriminant, having unit variance, and matrix $\mathbf{Y}$ with zero covariances. Therefore, the rule for classification using the appropriate measure of square distance and a subset of ℓ eigenvectors follows: allocate x to class k if $\forall_{i \neq k}$,

$$\sum_{j=0}^{\ell-1} (e_j(x - \overline{x}_k))^2 \leq \sum_{j=0}^{\ell-1} (e_j(x - \overline{x}_i))^2. \tag{3.20}$$

3.6.2 Classification results and discussion

The twenty randomly sized and spaced rectangular cuts were made from each of the 112 Brodatz texture images to produce 2240 total texture cuts. The training sets $\mathcal{S}_R^{\mathcal{A}}$ was formed by adding $\mathcal{A}$ cuts from each class, $\mathcal{A} = \{2, 3, \ldots\}$. The remaining $\mathcal{B} = 10$ cuts from each class were assigned to test set $\mathcal{S}_T^{\mathcal{B}}$ to evaluate the texture classification performance. An example of classification using the Brodatz texture experiemntal set-up is illustrated in Figure 3-10 and 3-11.

We examine several important parameters of the texture energy feature sets and the FDA process that impacts the texture classification performance.

3.6.2.1 Effect of the training class size

As the number of texture cuts from each class used for training increases, the classification rate increases, as indicated in Figure 3-12. As more training cuts are used, FDA produces more accurate discriminant functions. Good performance is found by using as little as three training textures per class.

3.6.2.2 Effect of the number of training classes

Figure 3-13 illustrates that the classification performance degrades only slightly when a quarter of the classes are used for training. In the image database application all possible texture classes are not available to derive the discriminant functions. Therefore, it is beneficial when training from only the available textures generates discriminant functions that perform well in classifying all textures.

3.6.2.3 Effect of the number of discriminant functions

The discriminant functions that account for the largest separation are indicated by having the largest normalized eigenvalues. By using only a subset of the discriminant functions that sufficiently separates the training classes, the computation for classification can be reduced.

It is advantageous to find feature sets which allow for the highest energy packing possible for classification. Figure 3-14 shows the results of classification, using most significant subsets of the discriminant functions. By using only the 5 to 10 most significant discriminants, the peak classification performance is reached.

3.6.2.4 Effect of the number of features

When correlated features are added to the feature set, FDA removes redundancy and produces a smaller set of uncorrelated discriminant functions. In general, any feature that provides a measure of class separation may be included in the feature set. However, there is a trade-off between classification performance and the increased computation as more features are added.

In this experiment, both mean absolute value (L_1 energy) and variance (L_2 energy) of the subbands are used in the feature sets. Figure 3-15 shows that the overall classification rate increases by using both energy measures.

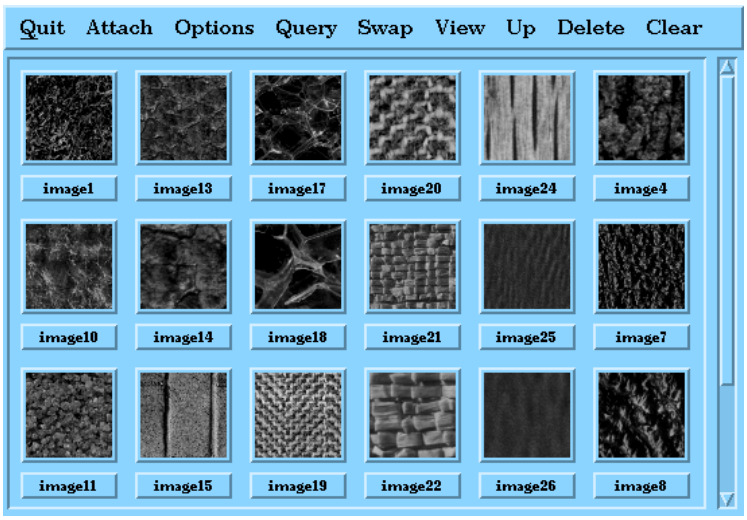


Figure 3-10: Brodatz texture classification experiment, texture cuts.

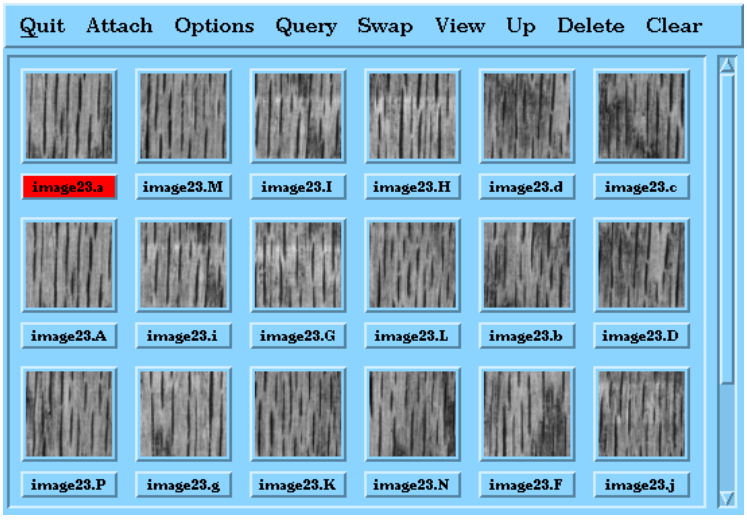


Figure 3-11: Brodatz texture classification experiment, retrieved texture cuts that best match query texture cut.

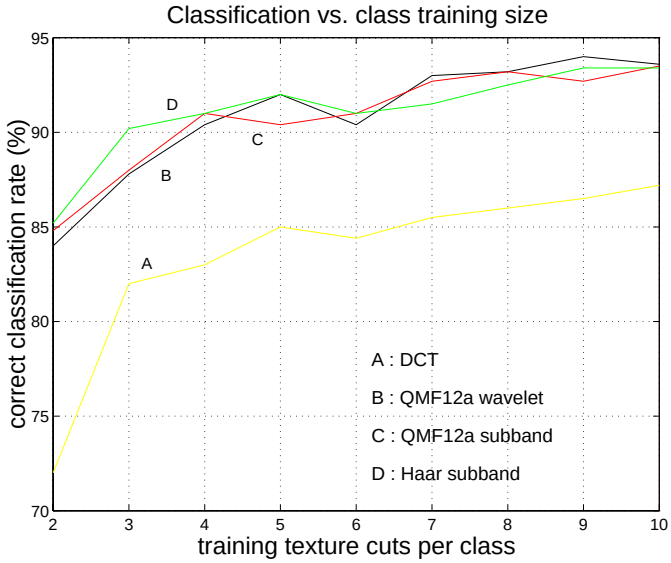


Figure 3-12: Effect on classification rate of the class size.

Figure 3-13: Effect on classification rate of the number of training classes.

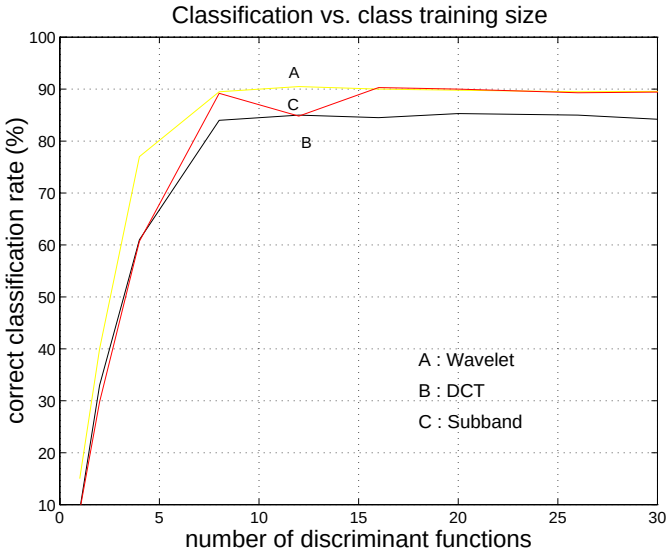


Figure 3-14: Effect on classification rate of the number of discriminant functions used for classification.

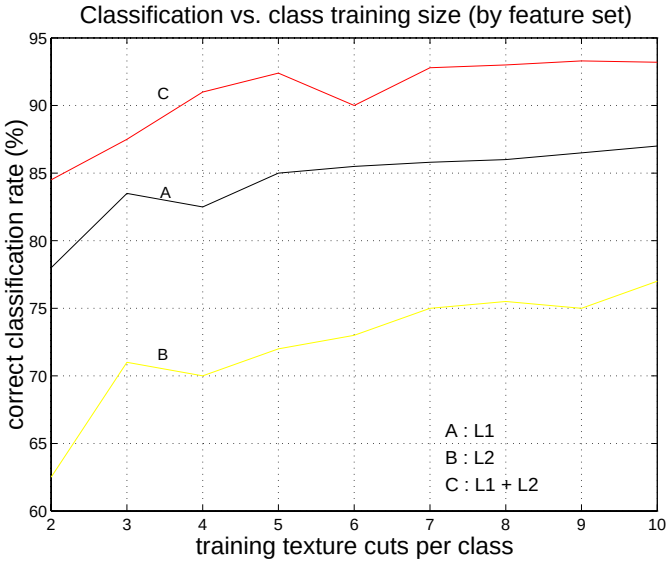


Figure 3-15: Effect on classification rate of the number of feature set.

3.7. Texture element histogram

We now use the texture energy features to derive the representation of texture using texture histograms. The texture histograms are generated in an overall process that consists of the wavelet filter back (FB), a texture channel generator (TCG), transformation T_t, and quantization Q_t as illustrated in Figure 3-16.

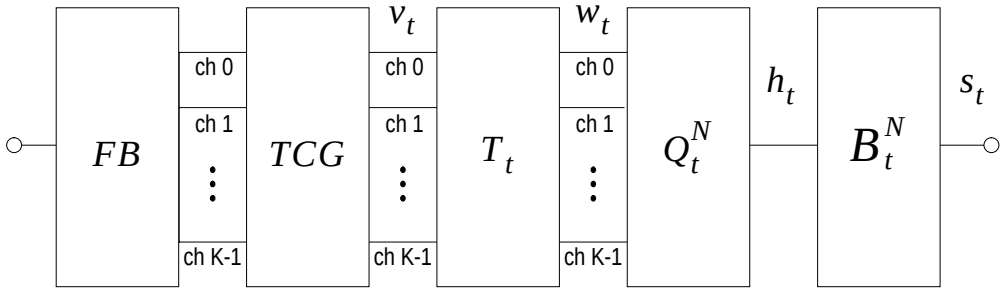


Figure 3-16: Generation of texture feature set. The filter bank (FB) and texture channel generator (TCG) generate nine texture channels. Transformation T_t and quantization Q_t produce the texture feature space in which texture histograms $\mathbf{h}_t$ and binary texture sets $\mathbf{s}_t$ are defined.

3.7.1 Texture filter bank (FB)

The wavelet FB and texture channel generator (TCG) system is illustrated in Figure 3-17. The 2-D four channel FB is iterated twice on the lowest *s-f* subband. By discarding the DC subband, nine channel subbands are obtained $\mathcal{Y}_k$, where $k = \{0 \ldots 8\}$.

3.7.2 Texture channel generator (TCG)

From each subband $\mathcal{Y}_k$ a texture channel is generated by computing the energy of the subband, upsampling and filtering. First, the energy ϵ_k of each point in subband k is computed from

$$\epsilon_k[x, y] = |\mathcal{Y}[x, y]^2|. \tag{3.21}$$

Then, ϵ_k is upsampled back to the full size by inserting zeros. The missing points are then filled-in using block filters $B_{i,j}$ to obtain texture channel S_k. The block filters perform simple pixel replication, where

$$B_{i,j}[x, y] = \begin{cases} ij & 0 \le x < i, \;\; 0 \le y < j \\ 0 & \text{otherwise.} \end{cases} \tag{3.22}$$

For each point in the image, the process illustrated in Figure 3-17 generates an energy value in each of the nine channels [141]. This representation is now symmetric with the representation of color (Chapter 2). Whereas, each point in a color image has values in the three color channels, here, we assign each point in the image energy values in the nine *s/s-f* channels.

The overall conversion process for the image to multiple channels is illustrated in Figure 3-18. As shown, the subbands of the wavelet image are used to generate the nine texture channels. Since the wavelet expansion of texture gives $k = 9$ *s/s-f* channels, a *texture point* is a 9-D vector. A texture point is defined as follows:

Definition 1 Texture point, $\mathbf{v}_t$. *A texture point is a vector* $\mathbf{v}_t = (S_0, S_1, \ldots, S_{K-1})$ *in* $\Re^K$, *where* S_k *gives the energy in s/s-f channel* k.

By transformation (T_t) and quantization (Q_t) of the 9-D *s/s-f* texture space, the texture points are reorganized and grouped to create a collection of *texture elements*. A *texture element* is generated as follows: given the texture point $\mathbf{v}_t$ and the transform T_t, for each $\mathbf{v}_t$ let the $\mathbf{w}_t$ be obtained by $\mathbf{w}_t = T_t\mathbf{v}_t$.

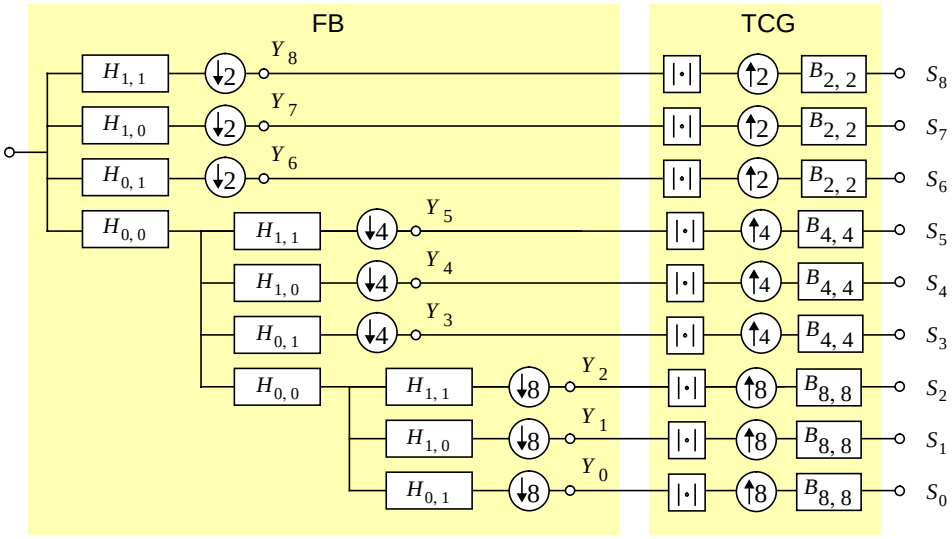


Figure 3-17: Texture filter bank (FB) and texture channel generator (TCG).

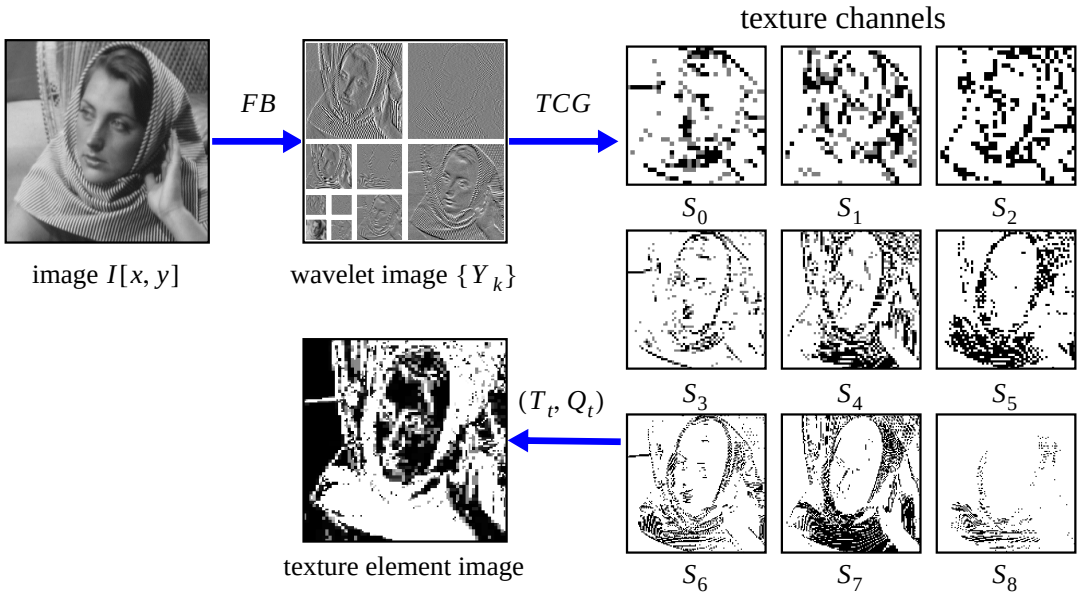


Figure 3-18: Overall process for representing texture using texture histograms.

Furthermore, let Q_t be a vector quantizer function (defined in Section 2.6.2) that maps the vector $\mathbf{w}_t$ to one of N codewords indexed by n. Then, n, where $n \in \{0 \ldots N-1\}$, is the index of texture element $\mathbf{w}_t$ assigned by the quantizer function and is given by $y_n = Q_t(\mathbf{w}_t)$, where y_n is the codeword with index n.

Definition 2 Texture element, *with index value n. A texture element is given by the complete set of texture points $\{\mathbf{v}_t\}_n$ that are assigned the same index n in the overall process of transformation and quantization: $y_n = Q_t(T_t\mathbf{v}_t)$.*

3.7.3 Texture space transformations

Under certain conditions it is desirable to obtain texture feature sets that are invariant to rotation or scaling. By transformation of the texture points, the features can be made approximately invariant to either rotation or scaling.

3.7.3.1 Rotation

The approximate rotation-invariance is obtained through the singular transformation T_t^R as follows

$$T_t^R = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{3.23}$$

The transformation T_t^R compensates for rotation by combining subbands at each scale. Since a rotated texture generates approximately the same scale energy as the un-rotated texture, T_t^R provides approximate rotation invariance. Figure 3-19 illustrates that the energy is redistributed in the texture channels during rotation. We see that the wavelet images change with rotation. Table 3.1 shows that the energy within the same scale is roughly constant in spite of the rotation of the textures. We evaluate the amount of rotation-invariance provided by examining the performance in classifying rotated Brodatz textures.

rotation (θ)	$S[0]$	$S[1]$	$S[2]$
6.11313°	205	1396	919
4.55157°	218	1247	874
−26.0922°	337	1327	893
−18.0881°	253	1471	993
−42.2369°	337	1183	792

Table 3.1: Rotated texture features. Energy is roughly constant within the same scale during rotation.

3.7.3.2 Rotated texture classification experiment

In a classification procedure similar that in Section 3.6., twenty randomly rotated, sized and spaced cuts were made from each of the 112 Brodatz texture images to produce 2240 total rotated textures. For training purposes, 10 cuts from each class are held aside. The additional 10 cuts from each class are set aside for evaluating the classification performance. The rotation transformation T_t^R is used to produce feature vector $\mathbf{w}_t$ for each rotated texture image

$$\mathbf{w}_t = T_t^R \mathbf{v}_t,$$

which are defined in 3-D space, $\mathbf{w}_t \in \Re^3$. The test set of $\mathbf{w}_t$'s are used in FDA to derive the discriminant functions for classifying the training set. Overall, the system obtains a correct classification rate of 52.32% for classifying rotated Brodatz textures using the transformation T_t^R of the texture energy feature sets. This represents a drop in the classification rate from that for unrotated textures (see Section 3.6.). However, considering the simplicity of the transformation T_t^R, the performance is acceptable.

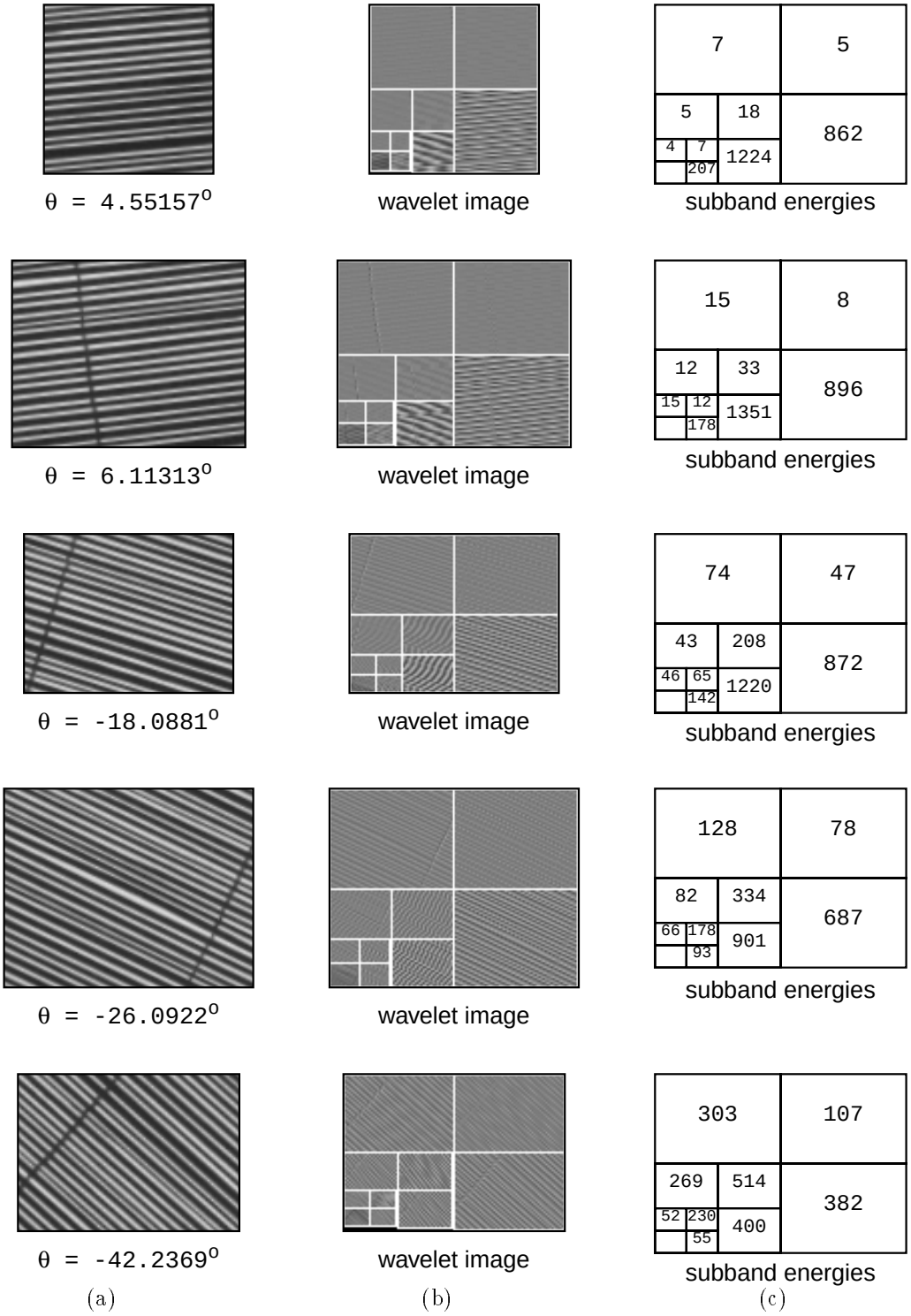


Figure 3-19: Rotated texture energy features: (a) several rotations of Brodatz texture d049 Straw screening, (b) wavelet images (c) L_2 subband energy values.

3.7.3.3 Scaling

The approximate scale-invariance is obtained through the singular transformation T_t^R as follows

$$T_t^S = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

The transformation T_t^S compensates for scaling by combining subbands across rotations. Since a scaled texture generates approximately the same rotation energy as the un-scaled texture, T_t^S provides approximate scale invariance. Figure 3-20 illustrates that the energy is redistributed in the texture channels during scaling. We see that the wavelet images change with scale. Table 3.2 shows that the energy within the same orientation is roughly constant in spite of the scale of the textures.

scale (ζ)	$S[0]$	$S[1]$	$S[2]$
2.211	282	49	339
2.118	275	75	414
1.582	365	82	456
1.264	429	96	570
0.986	503	159	596

Table 3.2: Scaled texture features.

3.7.3.4 Scaled texture classification experiment

In a similar classification procedure, twenty randomly scaled, sized and spaced cuts were made from each of the 112 Brodatz texture images to produce 2240 total scaled textures. The scaled transformation T_t^S was used to produce for feature vector $\mathbf{w}_t$ for each texture

$$\mathbf{w}_t = T_t^S \mathbf{v}_t,$$

which are defined in 3-D space, $\mathbf{w}_t \in \Re^3$. The test set of $\mathbf{w}_t$'s are used in Fisher Discriminant Analysis to derive the discriminant function for classifying the training set. Overall, the system obtained a correct classification rate of 61.82% for classifying scaled Brodatz textures using the transformation T_t^S of the texture energy feature sets. This represents a drop in the classification rate from that for unscaled textures (see Section 3.6.). However, as for transformation T_t^R, considering the simplicity of the T_t^S, the performance is acceptable.

3.7.4 Texture channel quantization

The next design parameter is the quantization Q_t of the texture channels. In order to produce a feature space for defining texture histograms and texture sets that is fairly compact, we quantize the texture energy channels. We investigate a procedure for choosing the quantizers to produce only two reconstruction levels for each channel. The procedure is easily extended to a greater number of reconstruction levels. We examine two quantizer design procedures – the Max-Lloyd and median trained quantizer design.

The Max-Lloyd quantizer for each channel is derived to minimize the mean square quantization error (MSQE) [126]. By treating the texture channel energy as a random variable, the goal of the quantization design is to produce mappings from texture channel energies to quantized values that minimize the error in quantization. The process of the quantization is illustrated in Figure 3-21. Given a quantizer input, represented by a random variable x, the output of the quantizer is $\hat{x}$. The quantization error is given by $\hat{x} - x$.

The MSQE-designed quantizer involves minimizing the squared quantization error, as follows

$$\epsilon = E[(x - \hat{x})^2] = \int_{d_l}^{d_u} (x - \hat{x})^2 p(x) dx, \tag{3.25}$$

where the input x ranges from d_l to d_u and $p(x)$ is the probability density function of x [126]. Given that x

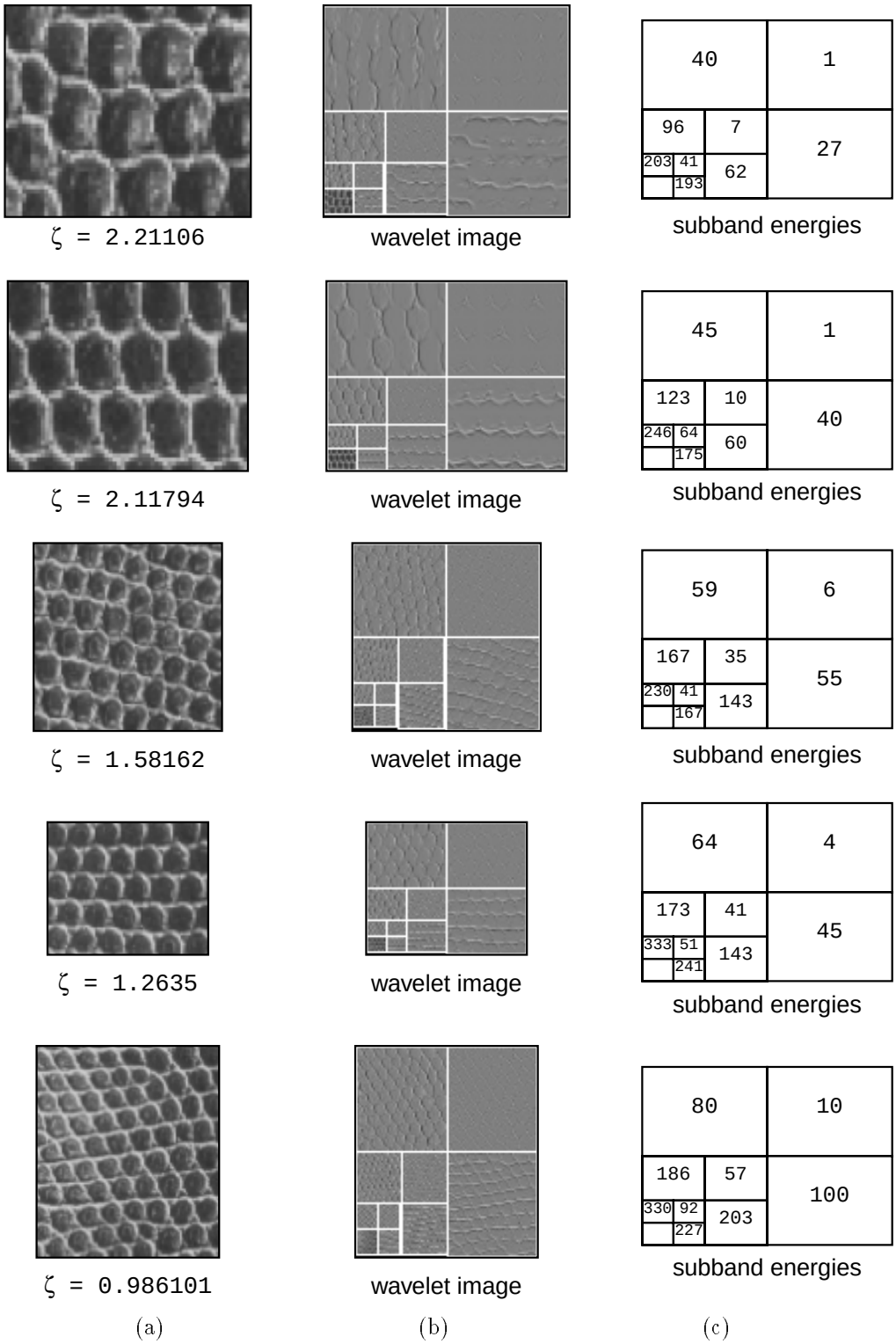


Figure 3-20: Scaled texture energy features: (a) several scalings of the Brodatz texture d035 Lizard skin, (b) wavelet images (c) L_2 subband energy values.

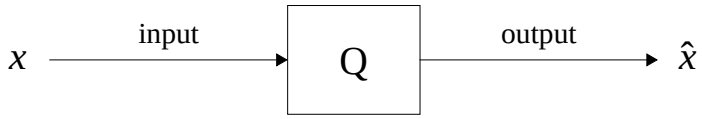


Figure 3-21: The quantizer Q produces output $\hat{x}$ for each input x. The quantization error is $\hat{x} - x$.

is a discrete random variable and $p[x]$ is its discrete-distribution, the error function is given by

$$\epsilon = E[(x - \hat{x})^2] = \sum_{x=d_l}^{d_u} (x - \hat{x})^2 p[x]. \tag{3.26}$$

The objective of the MSQE quantizer design is to choose a set of decision levels d_i and reconstruction levels r_i such that Eq 3.26 is minimized. A general tool for obtaining the sets of decision and reconstruction levels has been given independently by Max and Lloyd and is known as the Max-Lloyd quantizer. We use the results of the Max-Lloyd quantization design process to obtain the optimal decision and reconstruction levels for the texture channels.

3.7.4.1 Max-Lloyd texture quantizer design

The Max-Lloyd procedure finds the optimal MSQE for a given number J of output levels. Here, we consider $J = 2$, however, the procedure may be generalized to a greater number of reconstruction levels. We note that Eq 3.25 can be rewritten as

$$\epsilon = E[(x - \hat{x})^2] = \sum_{j=0}^{J-1} \int_{d_j}^{d_{j+1}} (x - r_j)^2 p(x) dx. \tag{3.27}$$

To minimize ϵ set $\frac{\partial \epsilon}{\partial r_k} = 0$,

$$\begin{aligned} \frac{\partial \epsilon}{\partial r_k} &= \frac{\partial \epsilon}{\partial} \left[\int_{d_{k-1}}^{d_k} (x - r_{k-1})^2 p(x) dx + \int_{d_k}^{d_{k+1}} (x - r_k)^2 p(x) dx \right] \\ &= (d_k - r_{k-1})^2 p(d_k) - (d_k - r_k)^2 p(d_k) = 0. \end{aligned} \tag{3.28}$$

The result gives that input level d_k is average of two adjacent output levels r_k and r_{k-1} as follows

$$d_k = \frac{r_k + r_{k-1}}{2}. \tag{3.29}$$

Since we are considering here the case that $J = 2$ (two output levels), we have to find only one decision level d_1. Since $d_1 = \frac{r_1 + r_0}{2}$, by finding r_0 and r_1 to minimize ϵ, the decision level d_1 is obtained. Therefore, we choose r_0 and r_1 to minimize the MSQE expression for $J = 2$ as follows

$$\epsilon = \sum_{x=0}^{d_1} (x - r_0)^2 p[x] + \sum_{x=d_1+1}^{\max(x)} (x - r_1)^2 p[x], \tag{3.30}$$

where $r_1 \geq r_0$ and $\max(x)$ is the upper bound on the input and $x = 0$ is the lower bound. To do this, we measure the distribution $p[x]$ of texture channel energies, and iterate through choices for r_0 and r_1, picking those that minimize ϵ.

3.7.4.2 Texture channel quantizers

We use the Brodatz textures to train the quantizers. We generate the texture channels using the texture FB and texture channel generation process described in Section 3.7.1. The texture channel quantizer training process is illustrated in Figure 3-22.

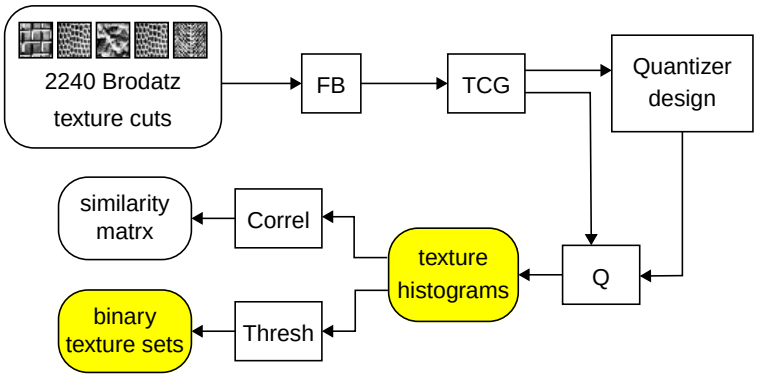
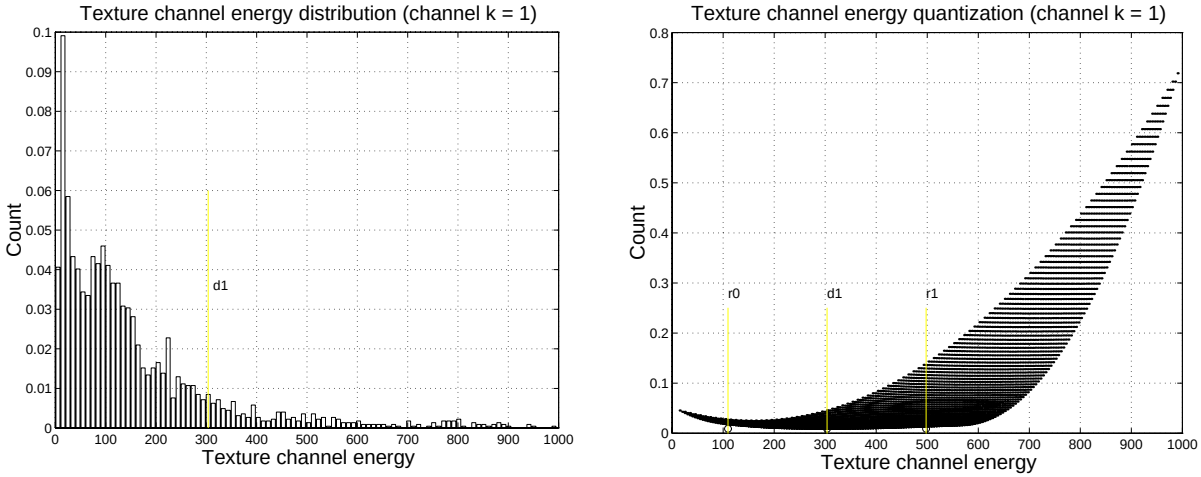


Figure 3-22: Texture channel quantizer training process.

Using the Max-Lloyd quantizer design process to obtain $J = 2$ output levels, an optimum decision threshold is determined for each channel. The histogram of texture channel energies for channel $k = 1$ is depicted in Figure 3-24(a). Figure 3-23(b) illustrates the optimal decision level d_1 and the reconstruction levels r_0 and r_1 for texture channel $k = 1$ based upon the sample set of Brodatz textures.



(a) (b)

Figure 3-23: Determination of optimal $J = 2$ output quantizer for texture channel $k = 1$.

The histogram of texture channel energies for channel $k = 8$ is depicted in Figure 3-24(a). Figure 3-24(b) illustrates the optimal decision level d_1 and the reconstruction levels r_0 and r_1 for texture channel $k = 8$ based upon the sample set of Brodatz textures.

By repeating this procedure on all nine channels, the optimal decision levels are given as follows:

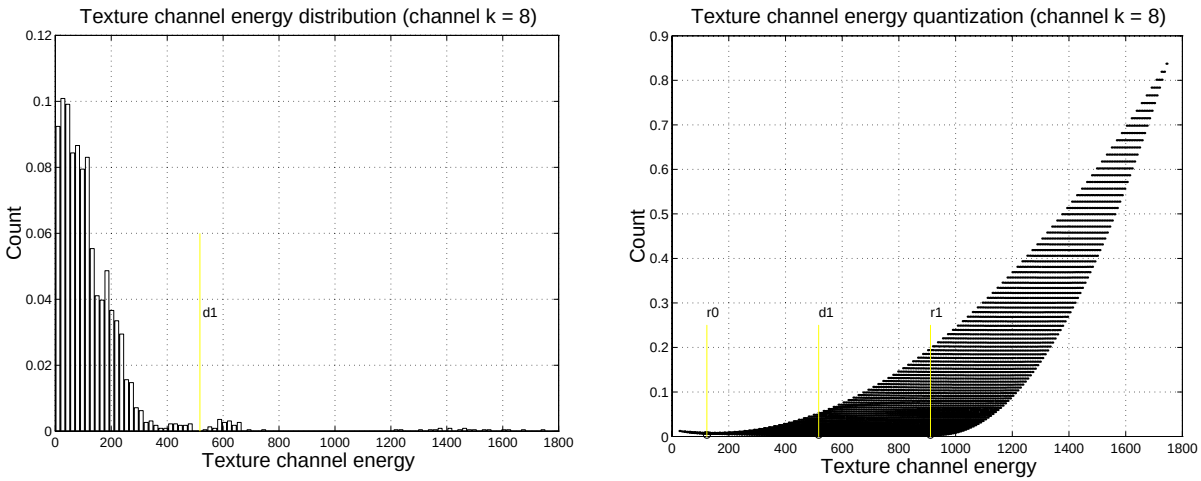
$$\Delta^*_{512} = (303.7, 357.3, 87.3, 642.3, 313.4, 60.4, 517.1, 682.7, 63.8).$$

3.7.4.3 Median texture energy quantizer

One alternative to the Max-Lloyd design procedure for choosing the optimal quantization thresholds is to simply threshold at the median of the texture channel energy distributions. By picking these thresholds, the optimal decision levels are given as follows:

$$\Delta^{\mathrm{m}}_{512} = (104.3, 113.4, 40.1, 120.6, 133.2, 33.3, 97.5, 113.4, 18.9).$$

Overall, since this gives two output levels for each of the nine channels, there is a combination of $2^9 = 512$



(a) (b)

Figure 3-24: Determination of optimal $J = 2$ output quantizer for texture channel $k = 8$.

outputs for the nine channels. From this 512-dimensional space, we define texture histograms and texture sets.

3.7.5 Texture histograms

The texture histograms are analogous to color histograms and are defined as follows

Definition 3 Texture histogram, $\mathbf{h}_c$. *A texture histogram determines the distribution of texture elements in an image, region or object.*

Given image $I[x, y]$ with nine *s/s-f* channels, S_k, $k \in \{0 \ldots 8\}$, the texture histogram is given by

$$h_t[n] = \sum_x \sum_y \begin{cases} 1 & \text{if } Q_t(T_t \mathbf{S}[x, y]) = n \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

The mean texture histograms obtained from 2240 Brodatz texture cuts are illustrated in Figure 3-25. In Figure 3-25(a) the texture channel quantization was derived from the Max-Lloyd procedure. In Figure 3-25(b) the texture channel qualtization was derived from the median texture channel energies.

3.7.6 Texture sets

The texture histograms are approximated by binary texture sets, which are defined as follows

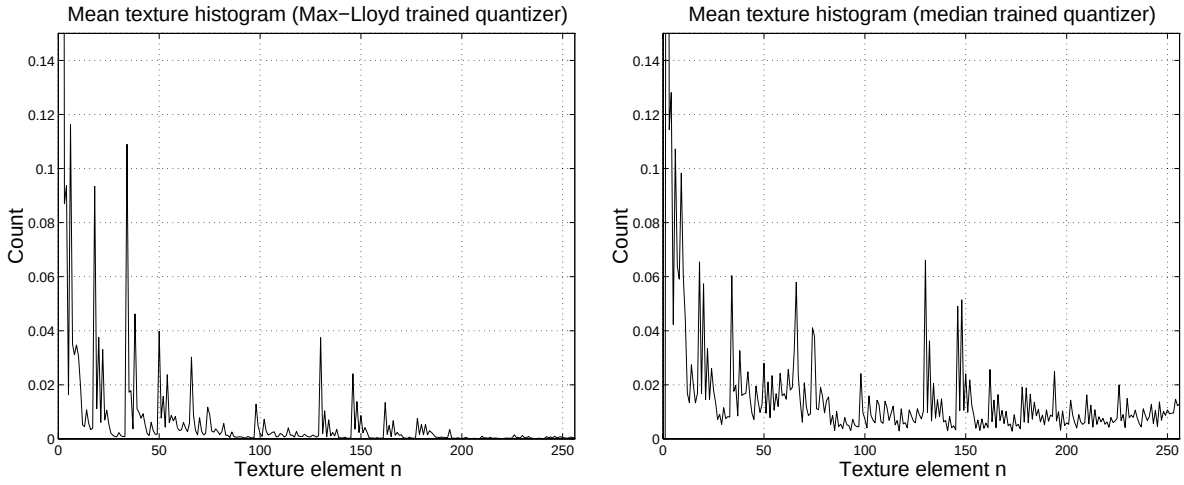
Definition 4 Texture set, $\mathbf{s}_c$. *A texture set is a binary vector in N-dimensional binary space $\mathcal{B}_t^N$ and determines a set of texture elements $\{n\}$ in an image, region or object.*

The mean binary texture sets obtained from 2240 Brodatz texture cuts are illustrated in Figure 3-26. In Figure 3-26(a) the texture channel qualtization was derived from the Max-Lloyd procedure. In Figure 3-26(b) the texture channel qualtization was derived from the median texture channel energies.

3.7.7 Texture set example

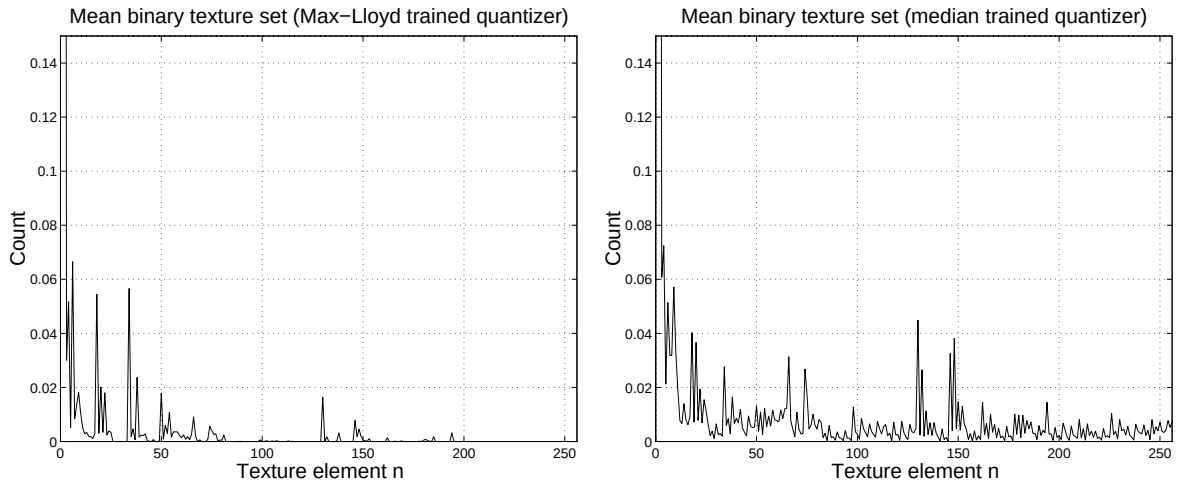
Figure 3-27(a) illustrates an example of the texture set representation using a $K = 3$ channel FB. The image is converted into the three texture channels (S_0, S_1 and S_2) using the FB and TCG. Next, Q_t quantizes each channel into two levels. The quantized texture elements are assigned an index n.

The overall system segments the texture regions in the images as illustrated in Figure 3-27(b). Given that each of the three channels is thresholded into two levels, eight texture elements are generated ($= 2^3$).



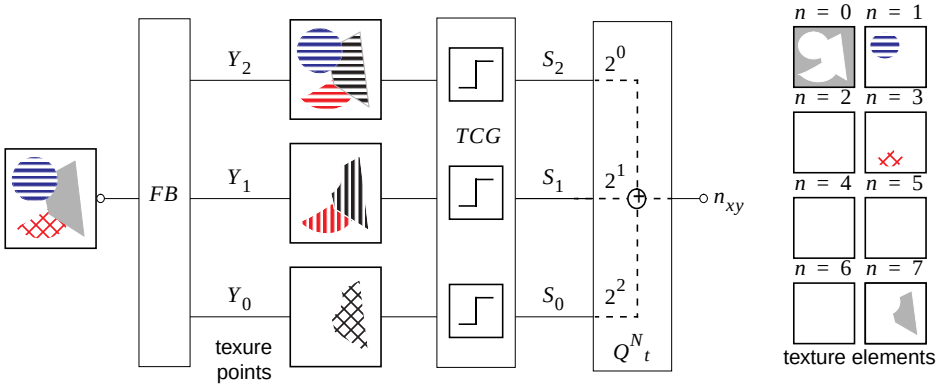
(a) (b)

Figure 3-25: Mean texture histograms from 2240 Brodatz texture cuts (a) obtained using Max-Lloyd trained texture channel quantizers, (b) obtained using median trained texture channel quantizers.



(a) (b)

Figure 3-26: Mean binary texture sets from 2240 Brodatz texture cuts (a) obtained using Max-Lloyd trained texture channel quantizers, (b) obtained using median trained texture channel quantizers.



(a) (b)

Figure 3-27: Example of texture set representation using a $k = 3$ channel FB.

3.8. Summary

In this chapter, we derived new representations of texture based upon histograms of texture elements. The texture elements are defined from the transformation (T_t) and quantization (Q_t) of QMF wavelet transform energies. We presented several experimental evaluations of the wavelet texture feature sets that demonstrate excellent performance in classifying Brodatz textures. We also investigated several transformations that allow for approximate rotation (T_t^R) or scale invariance (T_t^S).

The texture histogram representation satisfies the objectives of (1) providing a representation for image or regional texture that is compatible with color methods (namely, color histograms), (2) being compatible with compressed-domain texture extraction techniques (QMF wavelets), and (3) being suitable for region extraction techniques. In Chapter 5, we explore the process of extracting texture regions from images using texture histograms. First, in Chapter 4 we present and evaluate several techniques for computing texture similarity using the representations presented here.

By using the histogram and binary set representations of color (see Chapter 2) and texture we efficiently and effectively represent color and texture information in images and region. These visual dimensions are extremely useful for image search and retrieval. When combined with spatial information, the most powerful system for discriminating images is constructed (as we explore in Chapter 7). We now present techniques for measuring computing color and texture-based image queries using the histograms and binary sets.

Chapter 4

Feature Similarity and Indexing

4.1. Introduction

In this chapter, we investigate techniques for computing color and texture similarities using histograms. We also evaluate strategies for indexing the high-dimensional histograms. The objective of these techniques is to provide systems for search and retrieval of images by color and texture features.

We first evaluate the retrieval effectiveness of eight distance metrics for color and five distance metrics for texture in a series of image retrieval experiments. We evaluate these metrics in terms of precision and recall in retrieving images that are relevant to eight example image queries.

We next present two efficient strategies for computing histogram queries in large databases of images. The first, query optimized distance (QOD) computation, prioritizes the elements of the computation in the order of largest significance to the query. By prioritizing the computation in this way, the query is computed more efficiently. Furthermore, by stopping the computation process to only approximate the actual histogram distances, we show that the query response-time is reduced dramatically without significantly decreasing the retrieval effectiveness.

In the second method, binary set bounding (BSB), binary feature set queries are used to prefilter the more expensive color histogram queries. We formulate the BSB prefilter to provide no false dismissals. In this way, the complete histogram distance needs only to be computed for a fraction of the images in order to answer the histogram query. We now define the histogram metric space and the eight distance metrics which are suitable for feature-based image retrieval.

4.2. Histogram space

The features of color (see Chapter 2) and texture (see Chapter 3) are defined by histograms. The histograms depict the distribution of colors and texture elements, respectively, as they appear in images and/or regions. Since the histograms are discrete distributions, they can be represented as feature vectors in M-dimensional space, where M is the number of bins in the histogram. We define this $\Re^M$ space as the histogram space $\mathcal{H}^M$.

4.2.1 Histogram normalization

In order to compare regions and images of varying size, we introduce the following class of normalizations, where $r = 1, 2$,

$$\mathbf{h}^r = \frac{\mathbf{h}}{(\sum_{m=0}^{M-1} |h[m]|^r)^{1/r}}. \tag{4.1}$$

One effect of the normalizations in Eq 4.1 is a reduction in dimensionality of the feature space. Notice that in Figure 4-1 that the feature points for all two-bin histograms (normalized with $r = 1$) lie on a single line segment. That is, given one element of a histogram, the other is determined.

In general, an M-bin histogram normalized with $r = 1$ lies on an $M - 1$ dimensional simplex. An M-bin histogram normalized with $r = 2$ lies on the non-negative face of an M-dimensional hyper-sphere, which is illustrated for $M = 2$ in Figure 4-2.

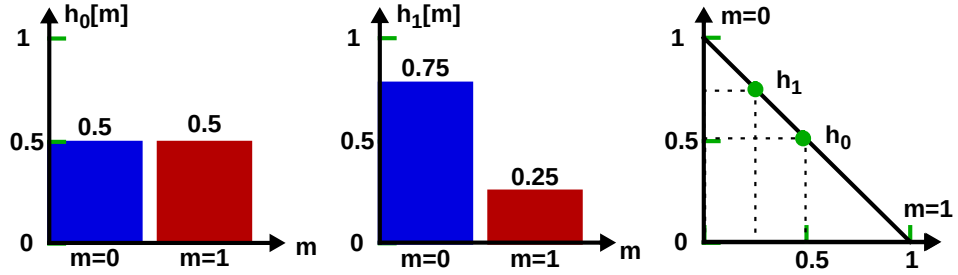


Figure 4-1: Feature vector representation of discrete-space, two-bin normalized histograms ($r = 1$).

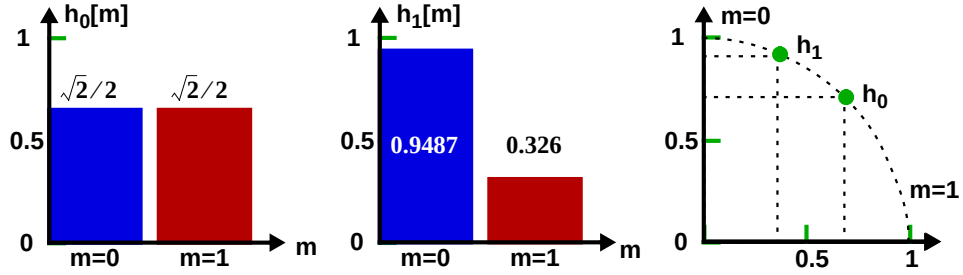


Figure 4-2: Feature vector representation of discrete-space, two-bin normalized histograms ($r = 2$).

4.2.2 Histogram metric space

The histogram space $\mathcal{H}^M$ is considered a metric space and the histograms $\mathbf{h}$ are points of the space if the following conditions hold: for every pair of histograms $\mathbf{h}_i, \mathbf{h}_j$ there can be found a corresponding number $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j)$, called the distance between points $\mathbf{h}_i$ and $\mathbf{h}_j$, which satisfies the following:

1. $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_i) = 0$ (identity),
2. $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) \geq 0$ (non-negativity),
3. $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) = \mathcal{D}(\mathbf{h}_j, \mathbf{h}_i) \geq 0$ (if $\mathbf{h}_i \neq \mathbf{h}_j$) (commutativity/symmetry),
4. $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_k) \leq \mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) + \mathcal{D}(\mathbf{h}_j, \mathbf{h}_k)$ (triangle inequality).

4.3. Taxonomy of distance metrics

In this section, we present and evaluate eight metrics ($\mathcal{D}1 \ldots \mathcal{D}8$) for measuring histogram dissimilarity, summarized in Table 4.1. The histogram metrics are evaluated along two dimensions: (1) retrieval effectiveness and (2) distance computation complexity. The second dimension includes the number of operations required in computing the distance scores. The first – retrieval effectiveness, provides a measure of the relevance of the histogram dissimilarity to subjective perceptual image dissimilarity. We evaluate in Section 4.4. the capability of the histogram distance metrics to retrieve images that are perceptually similar to query images.

4.3.1 Minkowski-form distance

The first class of dissimilarity measures is based upon the Minkowski-form distance metric. Let $\mathbf{h}_q$ and $\mathbf{h}_t$ be the query and target histograms, respectively, then

$$d^r_{q,t} = \left(\sum_{m=0}^{M-1} |h_q[m] - h_t[m]|^r\right). \tag{4.2}$$

As illustrated in Figure 4-3, histogram dissimilarity measures with the Minkowski-form ($\mathcal{D}1, \mathcal{D}2$, and $\mathcal{D}3$) neglect to compare similar, but not same, histogram elements. For example, a dark red image is equally

Metric	Description	Category
$\mathcal{D}1$	Histogram L_1 distance	Minkowski-form ($r = 1$)
$\mathcal{D}2$	Histogram L_2 distance	Minkowski-form ($r = 2$)
$\mathcal{D}3$	Binary set Hamming distance	Binary Minkowski-form ($r = 1$)
$\mathcal{D}4$	Histogram quadratic distance	Quadratic-form
$\mathcal{D}5$	Binary set quadratic distance	Binary Quadratic-form
$\mathcal{D}6$	Histogram Mahalanobis distance	Binary Quadratic-form
$\mathcal{D}7$	Histogram mean element distance	Non-histogram
$\mathcal{D}8$	Histogram moment distance	Non-histogram

Table 4.1: Summary of the eight histogram distance metrics ($\mathcal{D}1 \ldots \mathcal{D}8$).

dissimilar to a red image as to a blue image. By using measures of histogram element similarity within the distance computation, we investigate whether the quadratic metrics ($\mathcal{D}4, \mathcal{D}5$, and $\mathcal{D}6$) improve histogram matching.

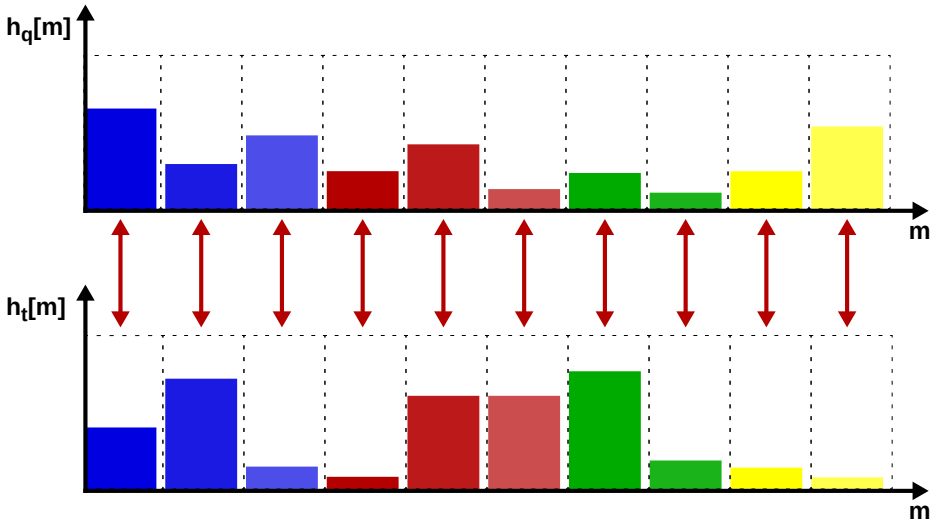


Figure 4-3: Minkowski-form distance metrics compare only "like" bins between the histograms.

4.3.1.1 Histogram intersection ($\mathcal{D}1$)

The histogram intersection was investigated for color image retrieval by Swain and Ballard in [159]. Their objective was to find known objects within images using color histograms. When the object (q) size is less than the image (t) size, and the histograms are not normalized, then $|\mathbf{h}_q| \leq |\mathbf{h}_t|$. The intersection of histograms $\mathbf{h}_q$ and $\mathbf{h}_t$ is given by:

$$d_{q,t} = 1 - \frac{\sum_{m=0}^{M-1} \min(h_q[m], h_t[m])}{|\mathbf{h}_q|}, \tag{4.3}$$

where $|\mathbf{h}| = \sum_{m=0}^{M-1} h[m]$. Eq 4.3 is not a valid distance metric since it is not symmetric: $d_{q,t} \neq d_{t,q}$. However, Eq 4.3 can be modified to produce a true-distance metric by making it symmetric in $\mathbf{h}_q$ and $\mathbf{h}_t$ as follows:

$$d'_{q,t} = 1 - \frac{\sum_{m=0}^{M-1} \min(h_q[m], h_t[m])}{\min(|\mathbf{h}_q|, |\mathbf{h}_t|)}. \tag{4.4}$$

Alternatively, when the histograms are normalized such that $|\mathbf{h}_q| = |\mathbf{h}_t|$, both Eq 4.3 and Eq 4.4 are true-distance metrics. It is shown in [159] that when $|\mathbf{h}_q| = |\mathbf{h}_t|$ that $\mathcal{D}1(q,t) = d_{q,t}$, and the histogram intersection is given by

$$\mathcal{D}1(q,t) = \sum_{m=0}^{M-1} |h_q[m] - h_t[m]|. \tag{4.5}$$

We recognize $\mathcal{D}1(q,t)$ as the Minkowski-form metric (Eq 4.2) with $r = 1$.

4.3.1.2 Histogram euclidean distance ($\mathcal{D}2$)

The euclidean distance is a Minkowski-form metric (Eq 4.2) with $r = 2$, as follows, given histograms $\mathbf{h}_q$ and $\mathbf{h}_t$,

$$\mathcal{D}2(q,t) = D2^2 = (\mathbf{h}_q - \mathbf{h}_t)^T(\mathbf{h}_q - \mathbf{h}_t) = \sum_{m=0}^{M-1}(h_q[m] - h_t[m])^2. \tag{4.6}$$

We decompose $\mathcal{D}2$ as follows

$$\mathcal{D}2(q,t) = \mathbf{h}_q^T\mathbf{h}_q + \mathbf{h}_t^T\mathbf{h}_t - 2\mathbf{h}_q^T\mathbf{h}_t. \tag{4.7}$$

When $|\mathbf{h}_q| = \mathbf{h}_q^T\mathbf{h}_q = 1$ and $|\mathbf{h}_t| = \mathbf{h}_t^T\mathbf{h}_t = 1$, we have

$$\mathcal{D}2(q,t) - 2 = -2\mathbf{h}_q^T\mathbf{h}_t. \tag{4.8}$$

In this decomposition, $\mathcal{D}2$ is computed from the inner product of the query $\mathbf{h}_q$ and target histograms $\mathbf{h}_t$. This decomposition has implications of efficiency in computing distances in the histogram database, which we explore in Section 4.5.4.

4.3.1.3 Histogram cosine distance

Closely related to the euclidean distance metric ($\mathcal{D}2$) is the cosine distance metric. The cosine metric is commonly used to measure the similarity of text documents [172]. The cosine metric computes the difference in direction, irrespective of vector lengths, where the distance is given by the angle between the two vectors.

Recall that the inner product between vectors $\mathbf{h}_q$ and $\mathbf{h}_t$ is given by

$$\mathbf{h}_q \cdot \mathbf{h}_t = \mathbf{h}_q^T\mathbf{h}_t = |\mathbf{h}_q||\mathbf{h}_t|\cos\theta.$$

Then, the cosine metric is given as

$$1 - \cos\theta = 1 - \frac{\mathbf{h}_q^T\mathbf{h}_t}{|\mathbf{h}_q||\mathbf{h}_t|}. \tag{4.9}$$

By substituting from Eq 4.6, where $|\mathbf{h}_q| = \mathbf{h}_q^T\mathbf{h}_q$ and $|\mathbf{h}_t| = \mathbf{h}_t^T\mathbf{h}_t$, we have

$$1 - \cos\theta = 1 - \frac{|\mathbf{h}_q| + |\mathbf{h}_t| - \mathcal{D}2}{2|\mathbf{h}_q||\mathbf{h}_t|}.$$

When $|\mathbf{h}_q| = |\mathbf{h}_t| = 1$, the cosine metric reduces to

$$1 - \cos\theta = \frac{\mathcal{D}2}{2}$$

.

Figure 4-4 illustrates the relationship between the cosine metric and the euclidean distance metric $\mathcal{D}2$. The cosine metric is not influenced by vector length. It gives the same distance result between $\mathbf{h}_0$ and $\mathbf{h}_1$ in Figure 4-4(a) as that between $\mathbf{h}_0$ and $\mathbf{h}_2$ in Figure 4-4(b). However, the euclidean distances differ in these cases, that is $\mathcal{D}2(\mathbf{h}_0, \mathbf{h}_1) \neq \mathcal{D}2(\mathbf{h}_0, \mathbf{h}_2)$.

4.3.1.4 Binary set Hamming distance ($\mathcal{D}3$)

We also consider the special case when the histograms are approximated by binary sets. Recall that we presented an algorithm for determining the nearest binary set to an arbitrary histogram in Section 2.8.4. A binary set $\mathbf{s}$ is a binary vector in the M-dimensional binary space, $\mathbf{s} \in \mathcal{B}^M$. The binary set Hamming distance between $\mathbf{s}_q$ and $\mathbf{s}_t$ is given by, where $|\mathbf{s}| = \sum_m s[m]$

$$\mathcal{D}3(q,t) = \frac{|\mathbf{s}_q - \mathbf{s}_t|}{|\mathbf{s}_q||\mathbf{s}_t|}. \tag{4.10}$$

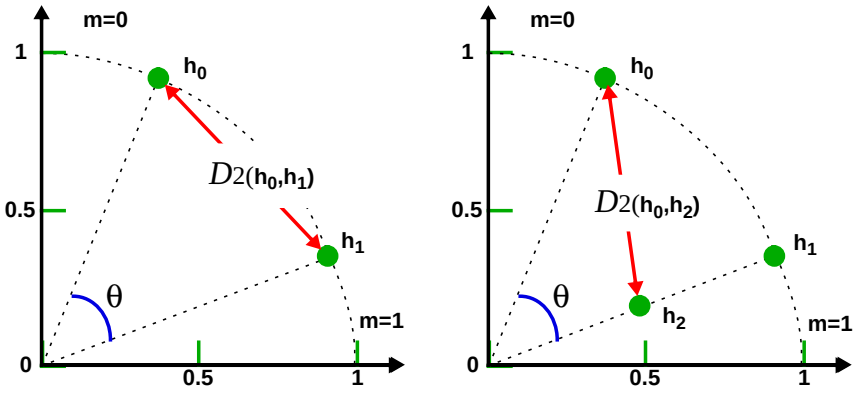


Figure 4-4: The cosine distance metric considers the angle θ between vectors, irrespective of vector lengths. The euclidean distance metric $\mathcal{D}2$ considers vector lengths and angle between them.

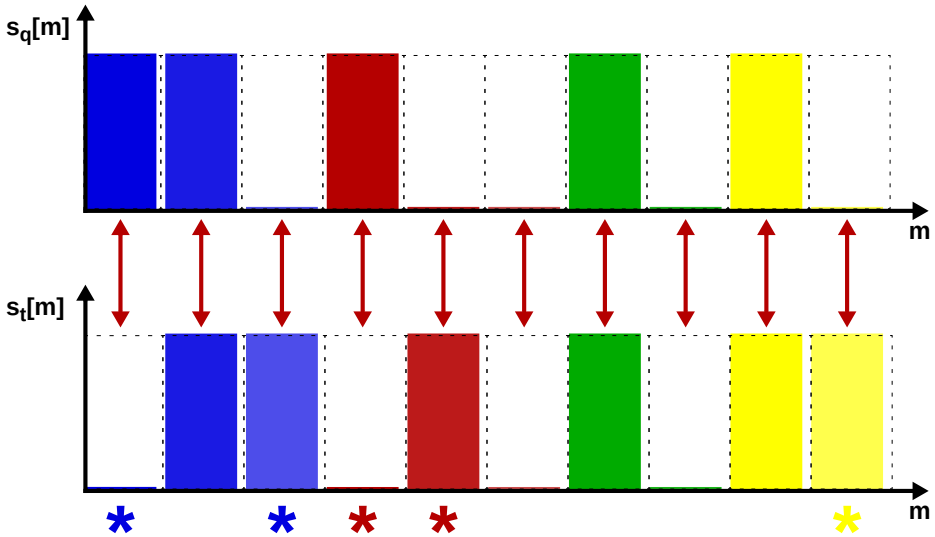


Figure 4-5: Binary set Hamming distance metric ($\mathcal{D}3$) computes the exclusive OR between elements. The '$\star$'s indicate the positions of bit difference.

Since the vectors are binary, the Hamming distance is determined by the bit-difference between the binary vectors [34]. As illustrated in Figure 4-5, $\mathcal{D}3$ is efficiently computed using an exclusive OR operator ($\odot$) which sets a one in each bit position where its operands have different bit values, and zero where they are the same [80],

$$\mathcal{D}3(q,t)|\mathbf{s}_q||\mathbf{s}_t| = \mathbf{s}_q \odot \mathbf{s}_t. \tag{4.11}$$

The binary set metric $\mathcal{D}3$ is efficient to compute, which justifies its exploration for use in large image database applications.

4.3.2 Quadratic-form distance

To address the shortcomings of the Minkowski-form distance metrics, we investigate three quadratic-form distance metrics: $\mathcal{D}4$, $\mathcal{D}5$ and $\mathcal{D}6$. The quadratic-form distance metrics compare all histogram elements, and weight the inter-element distance by pair-wise weighting factors (see Figure 4-6).

4.3.2.1 Histogram quadratic distance measures ($\mathcal{D}4$)

A quadratic-form distance metric is used in the IBM QBIC system for color histogram-based image retrieval [45]. In [102], it is reported that quadratic-form distance metric between color histograms provides more desirable results than "like-bin" only comparisons between color histograms. The quadratic-form distance between histograms $\mathbf{h}_q$ and $\mathbf{h}_t$ is given by:

$$\mathcal{D}4(q,t) = D4^2 = (\mathbf{h}_q - \mathbf{h}_t)^T \mathbf{A} (\mathbf{h}_q - \mathbf{h}_t), \tag{4.12}$$

where $\mathbf{A} = [a_{ij}]$ and a_{ij} denotes the similarity between elements with indexes i and j. The quadratic-form metric is a true-distance metric when $a_{ij} = a_{ji}$ (symmetry) and $a_{ii} = 1$.

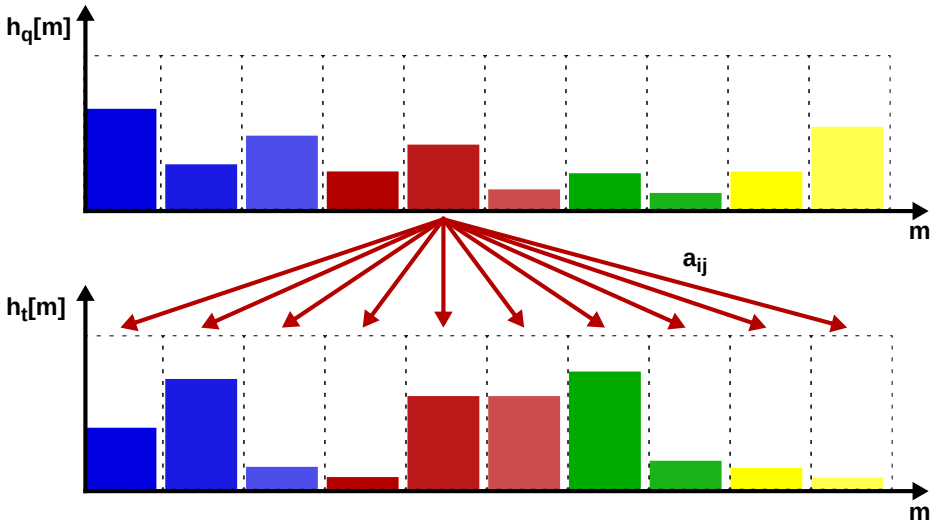


Figure 4-6: Quadratic-form distance metrics compare multiple bins between the histograms using similarity matrix $A = [a_{ij}]$.

In a naive implementation, the histogram quadratic distance is computationally more expensive than the Minkowski-form metrics since it computes the cross-similarity between all elements. We present, in Section 4.5.4, a strategy to decompose the quadratic-form metrics in a manner similar to that described for the $\mathcal{D}2$ metric (see Eq 4.8) to improve computation efficiency.

4.3.2.2 Binary set quadratic distance ($\mathcal{D}5$)

The quadratic-form distance metric also measures the distance between binary sets. The quadratic-form distance between two binary feature sets $\mathbf{s}_q$ and $\mathbf{s}_t$ is given by

$$\mathcal{D}5(q,t) = D5^2 = (\mathbf{s}_q - \mathbf{s}_t)^T \mathbf{A} (\mathbf{s}_q - \mathbf{s}_t), \tag{4.13}$$

By defining $\mu_q = \mathbf{s}_q^T \mathbf{A} \mathbf{s}_q$, $\mu_t = \mathbf{s}_t^T \mathbf{A} \mathbf{s}_t$ and $\mathbf{r}_t = \mathbf{A}\mathbf{s}_t$, and since $\mathbf{A}$ is symmetric, the quadratic-form binary set distance is reduced to the following expression

$$\mathcal{D}5(q,t) = \mu_q + \mu_t - 2\mathbf{s}_q^T \mathbf{r}_t. \tag{4.14}$$

4.3.2.3 Histogram Mahalanobis distance ($\mathcal{D}6$)

The Mahalanobis distance is a special case of the quadratic-form distance metric in which the transform matrix is given by the covariance matrix obtained from a training set of histograms, that is $A = \mathbf{\Sigma}^{-1}$. In order to apply the Mahalanobis distance, the histogram feature vectors are treated as random variables $\mathbf{X} = [x_0, x_1, \cdots, x_{M-1}]$. Then, the *correlation* matrix is given by R where $R = [r_{ij}]$ and $r_{ij} = \mathbf{E}\{x_i x_j\}$. Here, $\mathbf{E}\{y\}$ gives the mean of the random variable y. Then, the *covariance* matrix is given by $\mathbf{\Sigma}$, where $\mathbf{\Sigma} = [\sigma_{ij}^2]$ and $\sigma_{ij}^2 = r_{ij} - \mathbf{E}\{x_i\}\mathbf{E}\{x_j\}$.

The Mahalanobis distance between histograms is obtained by letting $\mathbf{X}_q = \mathbf{h}_q$ and $\mathbf{X}_t = \mathbf{h}_t$, which gives

$$\mathcal{D}6(q,t) = D6^2 = (\mathbf{X}_q - \mathbf{X}_t)^T \mathbf{\Sigma}^{-1} (\mathbf{X}_q - \mathbf{X}_t). \tag{4.15}$$

In the special case when x_i are statistically independent, but have unequal variances, $\mathbf{\Sigma}$ is a diagonal matrix as follows [163]

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_0^2 & & & 0 \\ & \sigma_1^2 & & \\ & & \ddots & \\ 0 & & & \sigma_{M-1}^2 \end{bmatrix}.$$

In this case, the Mahalanobis distance reduces to

$$\mathcal{D}6(q,t) = \sum_{m=0}^{M-1} \left(\frac{x_q[m] - x_t[m]}{\sigma_m}\right)^2. \tag{4.16}$$

When x_i are not statistically independent, a new coordinate system is obtained in which the components are statistically independent. This is implemented as a general eigenvalue problem whereby the goal is to obtain eigenvectors ϕ such that $\lambda\phi = \mathbf{\Sigma}\phi$. In this case, $\mathbf{X}$ is transformed using the eigenvectors such that $\phi\mathbf{Y} = \mathbf{X}$. Then, the Mahalanobis distance is given by

$$\mathcal{D}6(q,t) = \sum_{m=0}^{M-1} \frac{(y_q[m] - y_t[m])^2}{\lambda_m}. \tag{4.17}$$

4.3.3 Non-histogram distance

Finally, we examine two metrics ($\mathcal{D}7$ and $\mathcal{D}8$) which are derived from the image feature channels (i.e., color and texture) directly. However, the features are obtained from the histograms, which allows for the metrics to be defined from the histograms.

4.3.3.1 Feature channel mean distance ($\mathcal{D}7$)

The channel mean distance ($\mathcal{D}7$) is computed from the mean in each of the feature channels. For example, a mean color vector $\mathbf{v} = (\overline{r}, \overline{g}, \overline{b})$ is obtained by measuring the mean color in each of the three color channels (r, g, b). In this case, a suitable distance metric between mean color vectors $\mathbf{v}_q$ and $\mathbf{v}_t$ is given by

$$\mathcal{D}7 = D7^2 = (\mathbf{v}_q - \mathbf{v}_t)^T (\mathbf{v}_q - \mathbf{v}_t). \tag{4.18}$$

Note, that each bin in a color histogram $\mathbf{h}$ refers to a point (r, g, b) in the 3-D RGB color space. As such, $\mathbf{v}$ is computed from $\mathbf{h}$ by $\mathbf{v} = \mathbf{Ch}$, where $\mathbf{C}$ has size $M \times 3$, and $C[i]$ gives the (r, g, b) triple corresponding to histogram bin i. In general, for K feature channels and M histogram bins, $\mathbf{C}$ is has size $K \times M$. This allows us to rewrite Eq 4.18 as

$$\mathcal{D}7(q,t) = (\mathbf{h}_q - \mathbf{h}_t)^T \mathbf{C}^T \mathbf{C}(\mathbf{h}_q - \mathbf{h}_t). \tag{4.19}$$

4.3.3.2 Feature channel moment distance ($\mathcal{D}8$)

Similar to the case for the channel mean, other channel moments may be considered, such as variance and skewness. For example, color moments were explored by Stricker and Orengo for color image retrieval [158]. As in the case for $\mathcal{D}7$, these moments are also computed from the histograms. We define $\mathcal{D}8$ from the variance of each of the channels in order to compare the retrieval effectiveness of color moments to the other metrics.

4.3.4 Distance metric comparison

We now evaluate the performance of the distance metrics by conducting image retrieval experiments. We demonstrate that the set of metrics offers a trade-off in retrieval effectiveness and computational complexity.

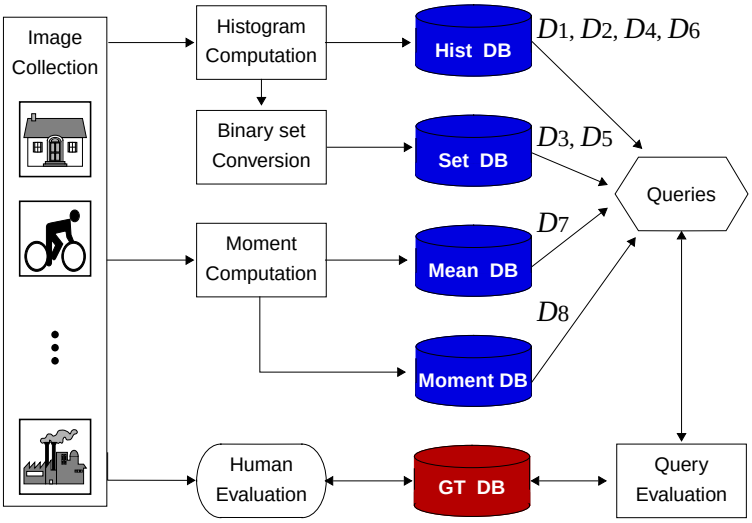


Figure 4-7: Distance metric retrieval effectiveness evaluation experiment setup.

4.4. Retrieval effectiveness evaluation

We first evaluate the histogram distance metrics by conducting image search experiments by color (eight metrics) and texture (five metrics). In the experiment setup, illustrated in Figure 4-7, we generate five image feature databases: (1) the histogram database, (2) the binary set database, (3) the mean element database, (4) the channel moment database, and (5) the ground-truth database. In the ground-truth database, the relevancies of each image to each of the queries is established manually. The retrieval effectiveness is evaluated by comparing the results of the queries using the feature databases to the ground-truths.

4.4.1 Image test set

Since no standard image corpus exists for the purpose of evaluating image retrieval methods, a test set of 3,100 color images was generated for this thesis. The image collection of 3,100 images was purchased on

CD-ROM from Expert Software*. Each of the 3,100 images was inspected and assigned a relevance to each of the following queries in order to measure the query's retrieval effectiveness. In this way, the image test set and assigned image relevance scores provide a common framework for comparing the image retrieval methods.

4.4.2 Retrieval effectiveness

Two metrics for retrieval effectiveness are *recall* and *precision*. Recall signifies the proportion of *relevant* images in the entire database that are *retrieved* in the query. Precision is the proportion of the *retrieved* images that are *relevant* to the query [76]. More precisely, let A be the set of relevant items, let B be the set of retrieved items, and a, b, c and d are given as follows.

- a = retrieved and relevant (detection)
- b = retrieved and not relevant (false alarm)
- c = not retrieved and relevant (miss)
- d = not retrieved and not relevant.

Recall and precision are defined by the following conditional probabilities:

$$\begin{array}{lcccccc} \text{recall} & = & P(B|A) & = & \frac{P(A \cup B)}{P(A)} & = & \frac{a}{a+c} \\ \text{precision} & = & P(A|B) & = & \frac{P(A \cup B)}{P(B)} & = & \frac{a}{a+b}. \end{array} \tag{4.20}$$

The measures of recall and precision require that the relevance to the query of each item in the database is established ahead of time. For the experiments, this was done for each query by subjectively assigning one of three values to each image in the database: relevant = 1, partially or marginally relevant = 0.5 and not relevant = 0. The relevance value corresponds to the probability that the item is relevant to the query.

In each query experiment, the distance (one of $\mathcal{D}1 \ldots \mathcal{D}8$) between each relevant image (with relevance = 1) and all images in the database is computed. The images are then sorted in order of lowest distance to the query image. In this order, the recall and precision are measured for each image.

The process is repeated for all the relevant images and an overall average retrieval effectiveness is computed for each distance metric and each query example. This produces just one recall *vs.* precision curve (i.e., see curve D1 in Figure 4-10). These experiments are repeated for all distance metrics to provide for comparison.

4.4.3 Color retrieval

We first evaluate the retrieval effectiveness using color histograms on the test set of 3,100 color images. The collection includes images with a variety of subjects, such as cities, nature, animals and transportation. We first examine example color queries: for images of (1) sunsets, (2) pink flowers, (3) nature scenes with blue skies and (4) lions.

4.4.3.1 Query 1: Sunsets

In Query 1, the goal is to retrieve images of sunsets. Prior to the trials, the 3,100 images were inspected, and 23 were designated to be relevant (to depict sunsets). Each of these images was used in turn to query the collection. This was repeated for each distance metric. Examples of the retrieved images and plots of the retrieval effectiveness are shown in Figures 4-10 and 4-11.

We see that all metrics are fairly successful at retrieving sunset images, except for $\mathcal{D}7$. Simply using the mean of each color channel does not sufficiently capture color information in the images. Table 4.2 compares the retrieval effectiveness at several operational values. We see that the quadratic-form metrics $\mathcal{D}4$ and $\mathcal{D}6$ perform well in Query 1. However, the retrieval effectiveness for $\mathcal{D}1$ is slightly better. Contrary to statements in [102], in Query 1 we do not see that the quadratic-form color histogram metrics improve performance substationally over the Minkowski-form metrics.

*Expert Software, Inc., 800 Douglas Rd., Coral Gables, FL 33134

23 sunset images	$\mathcal{D}1$	$\mathcal{D}2$	$\mathcal{D}3$	$\mathcal{D}4$	$\mathcal{D}5$	$\mathcal{D}6$	$\mathcal{D}7$	$\mathcal{D}8$
# relevant in top 10(20)	6(10)	5(9)	4(6)	5(9)	4(7)	5(9)	2(4)	3(5)
# retrieved to obtain 5(10) relevant ones	8(20)	9(24)	16(46)	9(22)	12(32)	8(21)	29(186)	18(56)

Table 4.2: Query 1: Sunsets. Comparison of eight distance metrics.

4.4.3.2 Query 2: Flowers

The goal of Query 2 is to retrieve images of pink flowers. Prior to the trials, the 3,100 images were inspected, and 21 were designated to be relevant (to depict pink flowers). Examples of the retrieved images and plots of the retrieval effectiveness are shown in Figures 4-12 and 4-13. Table 4.3 compares the retrieval effectiveness at several operational values. The performance of metrics $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}4$ and $\mathcal{D}6$ is similar. These all provide excellent image retrieval results, returning, on average, seven pink flower images among the first ten retrieved. The binary set Hamming distance metric $\mathcal{D}3$ provides a substantial drop in retrieval effectiveness. However, the binary set quadratic-form metric $\mathcal{D}5$ improves the retrieval effectiveness over $\mathcal{D}3$.

21 pink flower images	$\mathcal{D}1$	$\mathcal{D}2$	$\mathcal{D}3$	$\mathcal{D}4$	$\mathcal{D}5$	$\mathcal{D}6$	$\mathcal{D}7$	$\mathcal{D}8$
# relevant in top 10(20)	6(9)	6(9)	5(6)	6(9)	5(8)	7(10)	2(3)	3(3)
# retrieved to obtain 5(10) relevant ones	7(21)	7(24)	10(62)	7(21)	8(34)	7(18)	48(291)	43(185)

Table 4.3: Query 2: Flowers. Comparison of eight distance metrics.

4.4.3.3 Query 3: Nature

The goal of Query 3 is to retrieve images of nature with blue sky. Prior to the trials, the 3,100 images were inspected, and 42 were designated to be relevant (to depict nature with blue sky). Query 3 is the most difficult of the color image queries explored here. The test set contains many images with blue colors that are not from blue skies. Only the fraction of images that contain blue skies are deemed relevant. The other blue images are considered false alarms when retrieved by the image query methods.

We note that Query 3 is representative of a typical query for unconstrained images. We see that color information provides only a partial filter of image semantic content. However, we see in this query that the color query methods provide different performance in obtaining the images of semantic interest even in this difficult image query.

Examples of the retrieved images and plots of the retrieval effectiveness are shown in Figures 4-14 and 4-15. Table 4.4 compares the retrieval effectiveness at several operational values. Again, we see that the performance of metrics $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}4$ and $\mathcal{D}6$ is similar. These require that, on average, approximately 25 images need to be retrieved in order to obtain ten images that depict blues skies. The binary set Hamming distance metric $\mathcal{D}3$ provides a substantial drop in retrieval effectiveness, which is improved only little in the binary set quadratic-form metric $\mathcal{D}5$.

42 nature images	$\mathcal{D}1$	$\mathcal{D}2$	$\mathcal{D}3$	$\mathcal{D}4$	$\mathcal{D}5$	$\mathcal{D}6$	$\mathcal{D}7$	$\mathcal{D}8$
# relevant in top 10(20)	5(8)	5(8)	3(4)	5(8)	3(5)	5(9)	1(1)	2(3)
# retrieved to obtain 5(10) relevant ones	10(25)	10(26)	27(76)	10(26)	20(62)	9(23)	148(462)	33(108)

Table 4.4: Query 3: Nature with blue sky. Comparison of eight distance metrics.

4.4.3.4 Query 4: Lions

The goal of Query 4 is to retrieve images of lions. Prior to the trials, the 3,100 images were inspected, and 41 were designated to be relevant (to depict lions). Examples of the retrieved images and plots of the retrieval effectiveness are shown in Figures 4-16 and 4-17. Table 4.5 compares the retrieval effectiveness at several operational values. The performance of metrics $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}4$ and $\mathcal{D}6$ is excellent. Using these metrics, on

average, only 13 to 14 images need to be retrieved in order to obtain ten lions images. With $\mathcal{D}3$ and $\mathcal{D}5$, over twenty images must be retrieved.

41 lion images	$\mathcal{D}1$	$\mathcal{D}2$	$\mathcal{D}3$	$\mathcal{D}4$	$\mathcal{D}5$	$\mathcal{D}6$	$\mathcal{D}7$	$\mathcal{D}8$
# relevant in top 10(20)	7(14)	7(14)	6(9)	7(14)	6(9)	7(13)	5(8)	7(10)
# retrieved to obtain 5(10) relevant ones	6(13)	6(14)	8(26)	6(14)	7(22)	6(14)	8(32)	7(18)

Table 4.5: Query 4: Lions. Comparison of eight distance metrics.

4.4.3.5 Overall color retrieval evaluation

In Query 1 through Query 4, we compared the retrieval effectiveness of eight color query methods. We found that the color histogram query methods provide consistently good performance, regardless of the histogram distance metric used: $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}4$, or $\mathcal{D}6$. We also found that the quadratic-form metrics $\mathcal{D}4$ and $\mathcal{D}6$ provide little difference in retrieval effectiveness from non-quadratic-form metrics $\mathcal{D}1$ and $\mathcal{D}2$. This implies that the simplest metric ($\mathcal{D}1$) is a good candidate for use in image retrieval applications. These queries also evaluated the binary color sets. We found that the performance drops with binary sets. However, the simplicity afforded by these representations and metrics ($\mathcal{D}3$ and $\mathcal{D}5$) make them viable. Finally, the simple metrics based upon color channel measurements ($\mathcal{D}7$ and $\mathcal{D}8$) performed consistently poorly and are not sufficient for retrieving images by color features.

4.4.4 Texture retrieval

Next, we evaluate texture retrieval effectiveness using the Brodatz textures [9]. In the Brodatz experiment, we evaluate five distance metrics: $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{D}3$, $\mathcal{D}4$ and $\mathcal{D}5$. With metric $\mathcal{D}4$, the similarity matrix A is derived from the feature correlation matrix obtained by training on 1120 Brodatz texture cuts. We note that the we investigated $\mathcal{D}6$, the Mahalanobis distance, in the classification of texture in Chapter 3. We also explored, in Chapter 3, the metrics based upon the texture channels directly, and therefore, investigations of $\mathcal{D}6$, $\mathcal{D}7$ and $\mathcal{D}7$, are not repeated for texture image retrieval.

4.4.4.1 Query 5: Brodatz textures

In the Brodatz texture retrieval experiment, twenty random cuts (of fixed size, 96×96) were made from each of the 112 Brodatz textures (of size 512×512) to produce a total $2,240$ texture images. In each of the following query experiments, all twenty cuts from a class were used to query the collection of $2,240$ Brodatz texture cuts. The average retrieval effectiveness was computed from the twenty queries. We repeated each set of queries using each of the five texture distance metrics.

Query 5.1: Brodatz texture (Brodatz #D4: Pressed cork) In this experiment, images of pressed cork were used to query the Brodatz collection. The query results are shown in Figure 4-18. We see that texture histograms perform better than the binary texture sets. Table 4.6 compares the retrieval effectiveness at several operational values. The histogram metrics $\mathcal{D}1, \mathcal{D}2$, and $\mathcal{D}4$ perform about equally well. These return, on average, eight relevant images among the first ten retrieved. The binary texture sets return only five to six relevant images among the top ten retrieved.

20 pressed cork texture images	$\mathcal{D}1$	$\mathcal{D}2$	$\mathcal{D}3$	$\mathcal{D}4$	$\mathcal{D}5$
# relevant in top 10(20)	8(15)	8(15)	5(9)	8(15)	6(11)
# retrieved to obtain 5(10) relevant ones	6(12)	6(12)	9(22)	6(13)	7(17)

Table 4.6: Query 5.1: Brodatz texture (Brodatz #D4: Pressed cork). Comparison of five distance metrics.

Query 5.2: Brodatz texture (Brodatz #D1: Woven aluminum wire) In this experiment, images of woven aluminum wire were used to query the Brodatz collection. The query results are shown in Figure 4-19. We see that in this experiment, texture histograms perform much better than the binary texture sets.

In Query 5.2, the binary set representation does not discriminate the "woven aluminum wire" textures from the other textures. The reason is that many of the Brodatz textures, including "woven aluminum wire," have little energy outside of the DC subband and produce identical (empty) binary texture sets. Table 4.7 compares the retrieval effectiveness at several operational values. We found that $\mathcal{D}1$ and $\mathcal{D}4$ require only twelve images to be returned in order to retrieve ten relevant images. In contrast, $\mathcal{D}2$ requires 16 images to be returned.

20 woven aluminum wire images	$\mathcal{D}1$	$\mathcal{D}2$	$\mathcal{D}3$	$\mathcal{D}4$	$\mathcal{D}5$
# relevant in top 10(20)	8(14)	7(11)	0	8(14)	0
# retrieved to obtain 5(10) relevant ones	6(12)	6(16)	> 500	6(12)	> 500

Table 4.7: Query 5.2: Brodatz texture (Brodatz #D1: Woven aluminum wire). Comparison of five distance metrics.

4.4.4.2 Overall texture retrieval evaluation

In Query 5.1 and Query 5.2, we compared the retrieval effectiveness of five texture query methods. We found that texture histograms perform extremely well for metrics $\mathcal{D}1$, $\mathcal{D}2$ and $\mathcal{D}4$. We found that the binary texture sets have a undesirable failure mode. When the texture does not contain sufficient spatial-frequency energy outside of the DC subband, the binary texture set is empty. Since several Brodatz textures produce empty binary texture sets, the binary texture set representation does not discriminate between them (as shown in Query 5.2). However, in other cases (i.e., Query 5.1), the binary texture set provides retrieval performance that is only slightly worse than the texture histogram methods.

4.4.5 Color and texture retrieval

We now compare the visual dimensions of color and texture in retrieving images from the test set of $3,100$ images. In Query 6 through Query 8, we compare histogram metrics $\mathcal{D}2$ and $\mathcal{D}4$ for color with those for texture. We evaluate the relative efficacy of color *vs.* texture in retrieving images of (1) coral, (2) wild cats, and (3) yellow flowers.

4.4.5.1 Query 6: Coral

The goal of this query is to retrieve images of coral (which come in a variety of colors). Prior to the trials, the $3,100$ images were inspected, and 189 were designated to be relevant (to depict coral).

Examples of the retrieved images and plots of the retrieval effectiveness using color are shown in Figures 4-20 and 4-21. We can see that none of the distance metrics perform well since color alone only partially discriminates among the coral images. We compare the performance of color and texture in Figure 4-22. We can see that texture discriminates between the coral images differently than color. On average, the color metrics give slightly higher retrieval effectiveness than the texture metrics. However, in application, both visual dimensions can be used in order to provide a more complete system for retrieving images.

4.4.5.2 Query 7: Wildcats

The goal of this query is to retrieve images of wildcats. Prior to the trials, the $3,100$ images were inspected, and 26 were designated to be relevant (to depict wildcats). Examples of the retrieved images and plots of the retrieval effectiveness using color are shown in Figures 4-23 and 4-24. The color histogram metrics clearly retrieve the wildcats better than binary color set metrics do. Since the wildcat images do not contain much color information, the color sets do not sufficiently represent the wildcat images. As a result, the binary color sets perform poorly in retrieving these images.

We compare the performance of color and texture in Figure 4-25. In this example, color better discriminates among the wildcat images than texture. However, the texture representations capture the texture information in the wildcat images, such as the spots on the wildcats' heads. These features enable the wildcat images to be retrieved effectively using the texture metrics.

4.4.5.3 Query 8: Yellow flowers

The goal of this query is to retrieve images of yellow flowers. Prior to the trials, the 3,100 images were inspected, and 34 were designated to be relevant (to depict yellow flowers). Examples of the retrieved images and plots of the retrieval effectiveness using color are shown in Figures 4-26 and 4-27. As in the earlier cases, the color histogram methods perform better than the binary color sets in retrieving the yellow flowers. However, many yellow non-flower images are returned and reduce the effectiveness of the color metrics.

We compare the color metrics to texture in Figure 4-28. Texture alone does not satisfactorily retrieve yellow flowers. However, texture does retrieve images of flowers of other colors, as depicted in Figure 4-28. This query illustrates that color and texture provide complementary representations of the flower images. The texture features characterize the spatial-frequency detail of the flowers. Color adds discrimination power in determining the colors in the different flower images.

4.4.5.4 Overall color *vs.* texture retrieval evaluation

In Query 6 through Query 8, we compared the retrieval effectiveness of color and texture. We found in these image retrieval experiments that color is more effective than texture. However, texture is still important. In the case of the wildcat images (Query 7), there is little color in the images. As a result, we found that the simple binary color set metrics could not discriminate among the wildcat images. However, the texture metrics were able to retrieve the wildcat images. In the cases when the desired images are defined by color (Query 8, yellow flowers), color performs much better than texture. However, given the objective of retrieving images of coral of many colors (Query 6), texture and color perform nearly equally well.

The next important dimension in evaluating the feature distance metrics is the computation required to answer the queries. We present some of the recent approaches for computing histogram queries such as prefiltering and dimensionality reduction, and present two new methods: binary set bounding (BSB) and query optimized distance (QOD) computation.

4.5. Feature indexing

There is great difficulty in indexing high dimensional feature vectors. For example, traditional indexing techniques, such as the R-tree [58] are not suited for high-dimensional vector spaces. In general, the objective of feature indexing techniques is to reduce the amount of data retrieval and computation in order to complete the similarity search. The overall optimization considers the following criteria (1) computation, (2) data retrieval (i.e., retrieved disk blocks), (3) misses and (4) false alarms. For example, in some situations, it is desirable to allow some false matches in order to reduce computation and data retrieval.

Several recent techniques, such as prefiltering and dimensionality reduction, have been proposed for indexing histograms for CBVQ. The prefiltering divides the similarity search into stages such that the computation required for evaluation at each subsequent stage increases and the number of candidate feature points decreases [59]. Alternatively, feature reduction techniques have been investigated to reduce the computation and data retrieval in a similarity search, while possibly allowing use of indexing structures.

We propose two new techniques to improve feature indexing: binary set bounding (BSB) and query optimized distance (QOD) computation. We describe these new techniques and compare them to other methods based upon prefiltering and dimensionality reduction.

4.5.1 Prefiltering

The objective of prefiltering is to design a distance metric which is inexpensive to compute and can act as prefilter, thereby, eliminating the necessity of computing the more expensive distance computation for many of the images. The problem is posed as follows: design distance metric D_{pre} such that

$$D_{\text{pre}}(Q,T) \leq D_{\text{exp}}(Q,T), \tag{4.21}$$

where D_{pre} is inexpensive to compute and D_{exp} is more expensive to compute. If $D_{\text{pre}}(Q,T)$ satisfies Eq 4.21, then false dismissals are prevented [83].

We now look at the possibilities of using, in tandem, the distance metrics proposed above in order to arrange a successful prefilter.

4.5.1.1 Quadratic distance bounding

One special case of prefiltering for quadratic-form distance metrics was proposed in [59]. The basic idea is to use distance metric $\mathcal{D}7$ as a prefilter for distance metric $\mathcal{D}4$. It is shown in [59] that for $\mathbf{C}$ (in Eq 4.19) and $\mathbf{A}$ (in Eq 4.12) with certain properties (namely, $\mathbf{A}$ is positive semi-definite) that $\mathcal{D}4 \geq \lambda\mathcal{D}7$ for some constant λ, which is derived from $\mathbf{C}^{-1}\mathbf{A}$.

In this case, given a query with a quadratic distance bound ϵ ($\mathcal{D}4 \leq \epsilon$), matches are first found such that $\mathcal{D}7 \leq \epsilon/\lambda$, which guarantees no false dismissals. Then, $\mathcal{D}4$ is computed on this match list to find the best matches.

A naive procedure for computing $\mathcal{D}4$ on a database of size N has a complexity $O(NM^2)$, where M is the dimensionality of the histograms. Quadratic distance bounding requires the computation of $\mathcal{D}7$ on a database of size N plus a computation of $\mathcal{D}4$ on the smaller subset. This gives the new overall complexity $O(NK + LM^2)$, where K is the dimensionality of the vectors used in $\mathcal{D}7$ and $L << N$, which is a significant reduction.

4.5.2 Binary set bounding (BSB)

We define a new efficient prefiltering technique based upon binary set bounding (BSB). We found in the image retrieval experiments (see Section 4.4.) that histogram queries using distance metric $\mathcal{D}2$ provide excellent retrieval effectiveness. However, there are several disadvantages with $\mathcal{D}2$. First, the histograms, which are M-dimensional vectors, require floating-point representation. A significant amount of storage space is needed for each histogram. This introduces an overhead in the query process in accessing the histograms from disk. Second, $\mathcal{D}2$ requires $O(NM)$ floating-point multiplications in order to compute the histogram distances, where N is the size of the database.

On the other hand, the binary set Hamming distance metric $\mathcal{D}3$ does not retrieve images as well as $\mathcal{D}2$ (see Section 4.4.). However, the binary sets are stored with only single-bit representation. An M-dimensional binary set requires only M bits of storage space. Also, the computation of $\mathcal{D}3$ is extremely simple; it is computed from the bit-difference between histograms.

From these observations, it is natural to try and use the binary set metric $\mathcal{D}3$ to prefilter the database to reduce the computations needed for histogram metric $\mathcal{D}2$. The BSB technique provides the strategy for prefiltering as follows:

1. the query histogram (h_q) is mapped to the nearest binary set (s_q), with error $\epsilon_q(h_q, s_q)$,
2. the target histograms (h_{t_k}) are mapped, off-line, to their nearest binary sets (s_{t_k}) , with errors $\epsilon_{t_k}(h_{t_k}, s_{t_k})$, and
3. the binary set distance is computed for all target images, $\mathcal{D}3(q, t_k)$.
4. Given the histogram query threshold $\mathcal{D}2(q, t_k) \leq \tau$, the primary binary set query selects from the database only those images where $\mathcal{D}3(q, t_k) - \epsilon_{t_k} \leq \tau + \epsilon_q$. This provides for no false dismissals since $\mathcal{D}2(q, t_k) \leq \mathcal{D}3(q, t_k) + \epsilon_{t_k} + \epsilon_q$ is given by the triangle inequality.
5. Finally, $\mathcal{D}2$ is computed only on the images that have survived the $\mathcal{D}3$ prefilter stage.

Overall, the BSB prefilter technique requires $O(M \log M + M + pM)$ operations, where $p << N$, to compute the query. The binary set fast map (BSFM) algorithm from Section 2.8.4 determines the nearest binary set to a histogram, which requires $O(M \log M + M)$ operations. The remaining computation $O(pM)$ comes from computing $\mathcal{D}2$ on the surviving p images after prefiltering by the efficient bit-difference metric $\mathcal{D}3$.

4.5.3 Dimensionality reduction

Another approach is to use dimensionality reduction techniques directly on the set of feature vectors. There are two methods: data-independent reduction and data-dependent reduction – which requires a set of training feature vectors. The objectives are to compact the feature vectors in order to reduce disk access, computation and enable the use of indexing techniques [44, 1, 114].

Techniques for dimensionality reduction for image retrieval include the following, which are evaluated in Figure 4-8,

- singular value decomposition ("A" = SVD),

- discrete cosine transform ("B" = DCT),
- principal component analysis ("C" = PCA) and
- histogram sorting – ("D" = HS).

We examine the energy-packing ability of these methods using the 3,100 color image test set. The plots in Figure 4-8 depict the mean transformed histograms, where the elements are sorted in order of largest value (only the first 40 dimensions of the 166 element space are depicted).

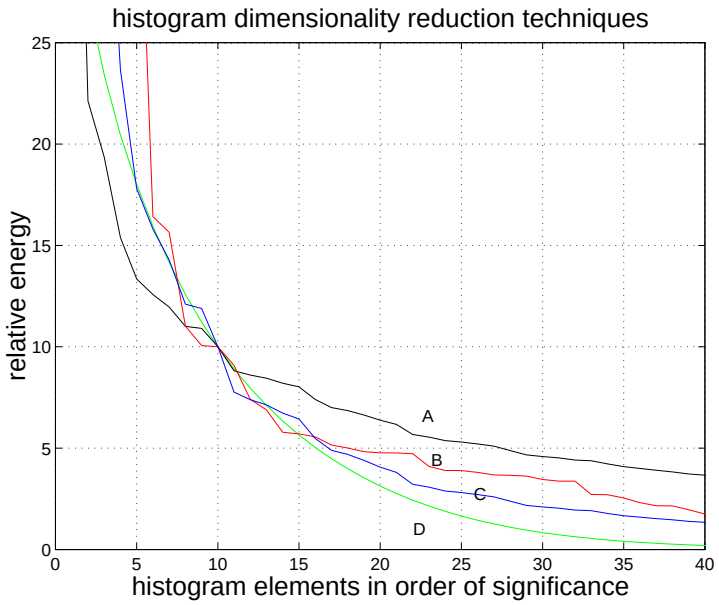


Figure 4-8: Results of histogram dimensionality reduction: "A" = DCT, "B" = SVD, "C" = PCA, "D" = HS (optimal sorting).

The plot for "A" gives the results for SVD – which obtains a unitary transformation of the histograms. We see that SVD compacts the histogram energy. The plot for "B" gives the results for DCT. In this case, the DCT of the histograms are first computed, then the elements of the mean of the DCT-transformed histograms are sorted in order of largest coefficients, giving "B." In this sense, we utilize a data-dependent form of the DCT. We see that the sorted-DCT trails-off faster than SVD.

The plot for "C" gives the results for PCA, which generates an eigenvalue decomposition of the histogram training data. We see that PCA compacts the histogram energy better than DCT and SVD. Finally, the plot for "D" is obtained by direct histogram sorting (HS), that is by sorting each histogram in order of largest element and then computing the mean of the optimal sorted training set. We see that HS compacts energy better than SVD, PCA and DCT methods.

The HS method is not practical for compacting the histograms for storage (we would need to store for each sorted-histogram also it's inverse sort transform). However, we use this discovery about histogram energy compaction from HS to derive a new powerful system for computing histogram queries – the query optimized distance (QOD) computation. The QOD computation method first sorts the query histogram (to obtain an energy compaction of the type depicted in Figure 4-8 – plot "D"), then accesses the elements of the target histograms in the database in order of largest value in the sorted query histogram.

4.5.4 Query optimized distance (QOD) computation

We present the query optimized distance (QOD) computation technique and show that it is applicable to both Minkowski-form (Section 4.3.1) and quadratic-form (Section 4.3.2) histogram and binary set distance metrics.

4.5.4.1 Quadratic-form QOD computation

We first explore the quadratic-form distance metric $\mathcal{D}4$ (Eq 4.12). By precomputing $\mu_t = \mathbf{h}_t^T \mathbf{A} \mathbf{h}_t$ and $\mu_q = \mathbf{h}_q^T \mathbf{A} \mathbf{h}_q$ and $\rho_t = \mathbf{A}\mathbf{h}_t$, we have

$$\mathcal{D}4 - \mu_q = \mu_t - 2\mathbf{h}_q^T \rho_t. \tag{4.22}$$

Let $\mathcal{M}_s$ be a permutation matrix that sorts $\mathbf{h}_q$ in order of largest element (HS). Applying this permutation also to ρ_t gives, where $\mathbf{f}_q = \mathcal{M}_s \mathbf{h}_q$ and $\theta_t = \mathcal{M}_s \rho_t$,

$$\mathcal{D}4 - \mu_q = \mu_t - 2\mathbf{f}_q^T \theta_t, \tag{4.23}$$

In this way, by first sorting the query histogram, the vector elements of ρ_t are accessed in order of decreasing importance to the query. It is simple to pre-compute ρ_t from the $\mathbf{h}_t$ for the database. We have now the following expression for computing $\mathcal{D}4$, where M is the dimensionality of the histograms,

$$\mathcal{D}4^{M-1} = \mathcal{D}4 - \mu_q = \mu_t - 2 \sum_{m=0}^{M-1} f_q[m]\theta_t[m]. \tag{4.24}$$

By stopping the summation at a value $k < M-1$, we obtain an approximation of $\mathcal{D}4^{M-1}$, such that

$$\mathcal{D}4^k \leq \mathcal{D}4^{k+1} \leq \ldots \leq \mathcal{D}4^{M+1}. \tag{4.25}$$

As such $\mathcal{D}4^k$ can be used in a process of bounding similar to that described above for quadratic distance bounding and BSB. Furthermore, the approximation of $\mathcal{D}4^k$ to the $\mathcal{D}4^{M-1}$ can be made arbitrarily close, and can be determined by the system or user in application.

This technique provides for a reduction of complexity in computing $\mathcal{D}4$. We have that nearly 80% of image color histogram energy is contained in the $k \approx 10$ most significant colors (see Figure 2-4). When only the most significant p colors are used, the complexity is $O(M \log M + kN + N)$. Here, $M \log M$ operations are required to sort the query color histogram and N is the size of the database.

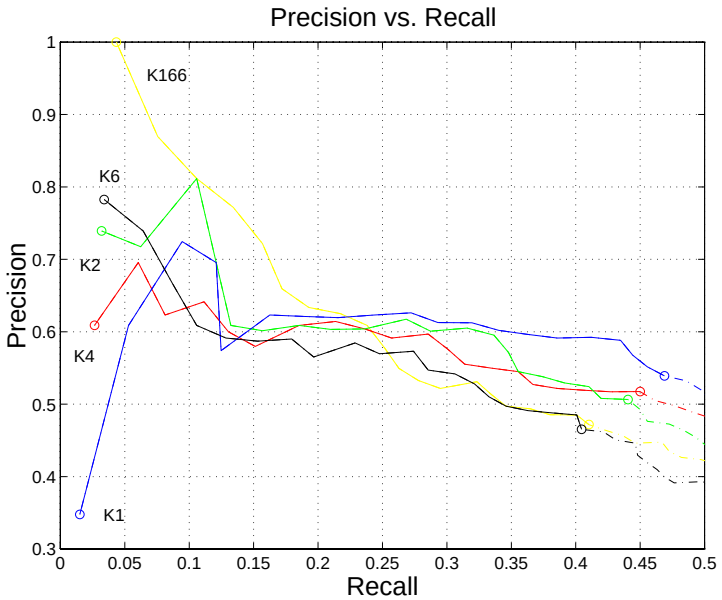


Figure 4-9: Average retrieval effectiveness for 23 queries for sunset images using QOD query method, K166 = all 166 colors, K6 = best 6 colors, K4 = best 4 colors, K2 = best 2 colors, K1 = best color.

To evaluate the impact of the QOD method on retrieval effectiveness, we repeat Query 1 (sunset images). In the experiment, we sort the elements of each query histogram bfh_q using HS. We then use only the k most significant colors ($k \in \{1, 2, 4, 6, 166\}$) to compute the query results. As in Query 1, we use each relevant image, in turn, to query the image test-set. The average retrieval effectiveness from the set of queries is given in Figure 4-9.

We make several observations from Figure 4-9. As k increases, the initial retrieval effectiveness also increases, which is expected. That is, the first images retrieved are more likely to be relevant. However, at the point where approximately ten images are retrieved the retrieval effectiveness starts to decline as k increases. We also see that for small values of k, i.e., $k = 6$, the retrieval effectiveness approaches that for $k = 166$. These results demonstrate that QOD provides a powerful method for reducing the computation required to answer the feature-based queries.

4.5.4.2 Binary set quadratic-form QOD computation

The QOD computation technique applies also to the binary set quadratic-form distance metric $\mathcal{D}5$ (Eq 4.14). Since $\mathbf{s}_q \in \mathcal{B}^M$ ($\mathbf{s}_q$ is a binary vector), we have, where $\mathbf{r}_t = \mathbf{A}\mathbf{s}_t$,

$$\mathcal{D}5(q,t) - \mu_q = \mu_t - 2 \sum_{\forall m, \text{ where } s_q[m]=1} r_t[m]. \tag{4.26}$$

Using this QOD computation, we need only to consider the elements of the query binary set where $s_q[m] = 1$ in the computation of the quadratic-form binary set distance $\mathcal{D}5(q,t)$.

4.5.4.3 Euclidean-form QOD computation

The euclidean-form QOD computation (of $\mathcal{D}2$) is a special case of the quadratic-form where $\mu_q = \mu_t = 1$ and $\rho_t = \mathbf{h}_t$. Since μ_t is constant for the query, the complexity is further reduced to $O(M \log M + pN)$.

4.6. Summary

In this chapter, we presented and evaluated techniques for computing color and texture similarities using histograms. We presented eight distance metrics and evaluated the retrieval effectiveness in experiments of retrieving images from a test set of images. In summary, we found that the non-quadratic-form metrics ($\mathcal{D}1$ and $\mathcal{D}2$) perform as well as the quadratic form metrics ($\mathcal{D}4$ and $\mathcal{D}6$). We also found that binary sets are an efficient and compact alternative to histograms. Retrieval effectiveness with binary set metrics ($\mathcal{D}3$ and $\mathcal{D}5$) is worse than histograms, but examples illustrate that their performance suffices as a first pass query.

We also compared the relative efficacy of the visual dimensions of color and texture in retrieving images from the test set. While texture histograms perform well in retrieving Brodatz texture images, color methods are better than texture methods in retrieving images from the set of "unconstrained" images.

We also presented two new strategies for efficiently computing the histogram distance queries using binary set bounding (BSB) and query-optimized distance (QOD) computation. We compared these techniques to quadratic distance bounding and other feature dimensionality reduction techniques. We note that these techniques may be combined to provide a powerful solution for efficiently computing the histogram queries.

In the next chapter, we examine the problem of extracting color and texture regions from unconstrained imagery. We use the color and texture histograms and binary feature sets in a novel process of region extraction via histogram back-projection. We demonstrate in Chapter 7 that by extracting the regions from images and by utilizing spatial and feature attributes in the query process, we improve retrieval effectiveness significantly over the purely feature-based techniques explored in this chapter.

Chapter 5

Region Extraction

5.1. Introduction

In this chapter, we present a novel system for detecting and extracting color and texture regions. We present the framework for color and texture histogram back-projection. Histogram back-projection (HBP) determines the most likely spatial locations of a given histogram within an image. We also define the procedure for binary set back-projection which identifies the most likely locations of a given binary set within an image. We investigate and compare five formulations of back-projection.

We also develop a procedure that automatically extracts regions from images by testing candidate binary sets by performing simple binary set back-projections. The binary set iteration (BSI) procedure automatically generates single- and multiple-element binary sets for an image in order to extract regions.

These powerful techniques allow us to define an overall strategy for the integrated spatial and feature query system which includes the automated analysis of the images and extraction of color and texture regions.

5.2. Visual feature extraction

In the image retrieval applications, it is impractical to perform the back-projection or, in general, region extraction, at the time of the query. Therefore, in order to develop the complete solution for integrated spatial and feature querying by regions, we consider first that prototypal color and texture patterns are available. Given a collection $\mathcal{N}$ with N images and a set $\mathcal{L}$ with N color and texture patterns, we back-project the L patterns onto the N images to generate a region database $\mathcal{R}$ with R regions. The process is illustrated in Figure 5-1.

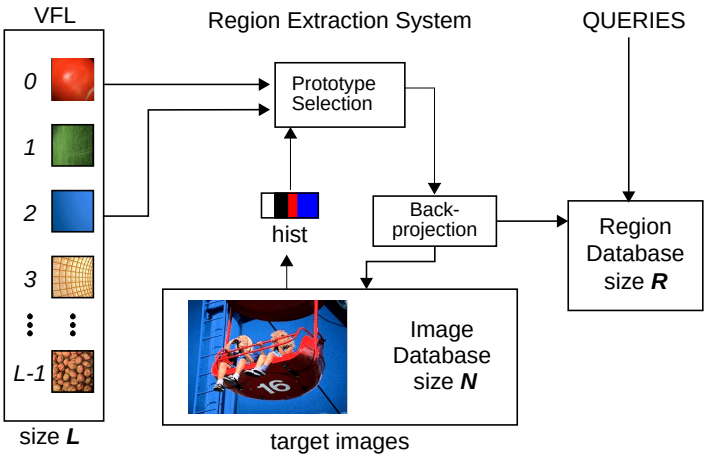


Figure 5-1: Strategy for region extraction in the integrated spatial and feature image query system.

The set of color and texture patterns $\mathcal{L}$ is generated in a number of ways. One method is to manually extract example regions from the images in the collection $\mathcal{N}$. Alternatively, fixed block segmentation of the

images combined with clustering provides examples (centroids) of the typical color and texture patterns in the image collection. These can be used to form $\mathcal{L}$. Finally, we present a novel approach in Section 5.5. for generating the set $\mathcal{L}$ automatically using binary feature sets.

The set of color and texture prototypes form the visual feature library (VFL) in which each element refers to a particular visual prototype of color, or texture, or both. Corresponding to each element $\mathcal{L}_k$ in the VFL is a feature type v_k (such as color or texture), a histogram $\mathbf{h}_k^v$, a histogram metric space $\mathcal{H}_k^v$, and a concept or class ω_k related to the prototype such that $\mathcal{L}_k = [v_k, \mathbf{h}_k^v, \mathcal{H}_k^v, \omega_k]$.

Whereas $\mathbf{h}_k^v$ – the measured feature, or histogram, is drawn from a multidimensional continuum of values, the variable ω_k comes from a discrete and finite set Ω of concepts or classes [133]. To each $\omega_k \in \Omega$ a distinct name is associated, i.e., "grass," "checker board," "sky," "zebra stripes."

The back-projection techniques detect in images, the elements $\mathcal{L}_k$, which are assumed to be given. In particular, the strategy for region extraction is as follows: for all elements $\mathcal{L}_k$ in the VFL, select each and try to detect (or to verify) it in the image. In this way, the strategy uses a top-down or model-driven approach to the general problem of recognition [104].

5.2.1 Region extraction strategies

There are many possible strategies for region extraction in CBVQ. The least complex method involves manual extraction. The images are evaluated by people, and the pertinent information is identified visually. For example, this approach is explored in the IBM QBIC system [102]. However, the manual extraction of regions and objects is extremely tedious and time-consuming for large image and video collections. Several recent techniques have been devised to facilitate the manual region extraction process, such as the use of active contours and snakes [78, 57] to automatically improve the coarse descriptions of region boundaries that are obtained manually.

Alternatively, blocks of regions can be extracted by fixed segmentation of the images. This approach was investigated by Stricker and Dimai for retrieving color images [157]. In their system, the images are divided into five regions. Using this type of segmentation, the authors improved image retrieval performance. However, in general, it is difficult to pick the scale and locations at which images should be blocked. Furthermore, fixed block segmentation is not invariant to shift or scale changes.

A third technique involves image segmentation. Many techniques have been proposed for segmenting images. Chua, Lim and Pung developed a segmentation technique which uses color pairs [29]. Later, Hsu, Chua and Pung extended the color pairs technique to perform foreground object color extraction [66]. Celenk developed an image segmentation technique which performs a clustering of colors [14]. The process determines simple structures and surfaces. Color histogram thresholding has been investigated extensively for segmenting color images. Ohta, Kanade and Sakai investigated one recursive histogram thresholding technique to derive a color space that is most suited for region segmentation [107]. However, the general problem of segmentation of unconstrained images is poorly defined, as we explain shortly.

In order to extract region from images that are used in the spatial and image feature query system, we propose a new technique which partly employs the HBP method developed by Swain and Ballard [159]. The authors use HBP to detect known objects within images [159]. HBP was later investigated by Ennesser and Medioni for devising a system which solves the "Find Waldo" puzzles ([33]) by computer [43].

5.2.1.1 Histogram back-projection

We define HBP as follows:

Definition 1 Histogram back-projection (HBP). *A class of algorithms designed to detect within images the regions with feature-histograms that are similar a given model histogram* $\mathbf{h}_q$.

In Swain and Ballard's formulation, HBP determines the most likely location of a specific histogram $\mathbf{h}_q$ within an image $I[x, y]$ [159]. By back-projecting the quotient of the query histogram and the image histogram, the location of the query histogram is found. More specifically, given query histogram $\mathbf{h}_q$ and image histogram $\mathbf{h}_t$, $\forall_{m \in 0 \ldots M-1}$ let $s[m] = \min(h_q[m]/h_t[m], 1)$. Then, replace each point in the image by the corresponding confidence score, $\forall_{x,y}$, $I'[x, y] = s[k]$, where $k = I[x, y]$. After convolving $I'[x, y]$ with a blurring mask, the location of the peak value corresponds to the most likely location of the model histogram within the image.

In small image retrieval applications, the HBP may be computed at the time of query to find objects within images [159, 43]. However, for large collections it is not feasible to compute the back-projection at time of query. A faster color indexing method is needed.

5.2.1.2 Region extraction in the image collection

We apply the back-projection technique to the problem of image retrieval by pre-computing the back-projections of elements $\{\mathcal{L}_k\}$ onto the images. By processing these back-projections ahead of time, the system solves the spatial and feature queries without extracting regions at the time of the query.

We begin by establishing the connection between the back-projection method and detection theory. We show that the back-projection techniques are closely related to those for pattern recognition and signal detection. However, we demonstrate that, since these approaches are overly constrained, they cannot be used as a system for region extraction in a collection of unconstrained images. We demonstrate that the back-projection coupled with the VFL provide a powerful solution for extracting regions in a large collection of unconstrained images.

We first describe the problems addressed by the general techniques of pattern recognition (PR) and image segmentation (IS). We argue that the frameworks presented for PR and IS are not suited to the problem of region extraction in the collection of unconstrained images because

1. pattern recognition requires that the set of region classes Ω is determined *a priori*, and is fixed, $|\Omega| = N$ for the duration of the application, and

2. the segmentation of unconstrained images is not a well defined problem.

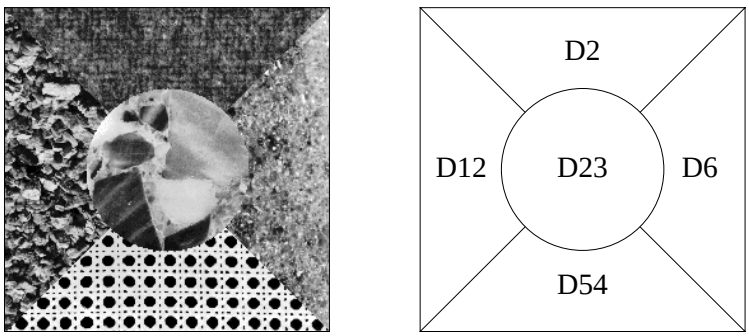
In general, it is not possible to determine the ground-truth segmentation from which the results of an image segmentation can be evaluated.

5.2.2 Pattern recognition

Pattern recognition is the process of inferring the class ω_k of an observation $\mathbf{h}_k^v$ [133]. There typically exists a finite set of K possible classes, $\Omega = \{\omega_0, \omega_1, \ldots, \omega_{K-1}\}$, where ω_k are the classes and Ω is the set of classes.

In a closed-world assumption, the K classes are mutually exclusive and complete [133]. Associated with each ω_k is a measurement vector $\mathbf{h}_k^v$, which is a collection of observations about class ω_k. In general, the measurement vector $\mathbf{h}^v$ defines a vector point in the M-dimensional space $\mathcal{H}^v$.

The objective of pattern recognition is to provide the mapping from the $\mathbf{h}_v$'s into the ω's through a decision function $\mathbf{d}(\mathbf{h}_v)$ as follows: $\mathbf{h}_v \xrightarrow{\mathbf{d}} \omega$.



(a) (b)

Figure 5-2: Pattern recognition is the process of classifying image points into a set of known classes (Ω). (a) Brodatz texture composite image, (b) example of labels (ω_k) assigned by pattern recognition process.

Typically, pattern recognition is accomplished by computing the distance of the unknown pattern to each class. Then, the unknown pattern is assigned to the nearest class k. This is accomplished as follows, given the unknown pattern with feature measure h^v, allocate h^v to class k if $\forall_{i \neq k}$, $\mathcal{D}^v(h^v, h_k^v) \leq \mathcal{D}^v(h^v, h_i^v)$, where $\mathcal{D}^v$ is the measure of dissimilarity of the type v features, (i.e., $v \in \{\text{color, texture}\}$).

Pattern recognition is not suited for the problem of extracting regions in a collection of unconstrained imagery because:

1. the set of classes Ω must be determined before hand, and has fixed size. For example, when a new pattern class is added, the prior results of a pattern recognition procedure are invalidated.

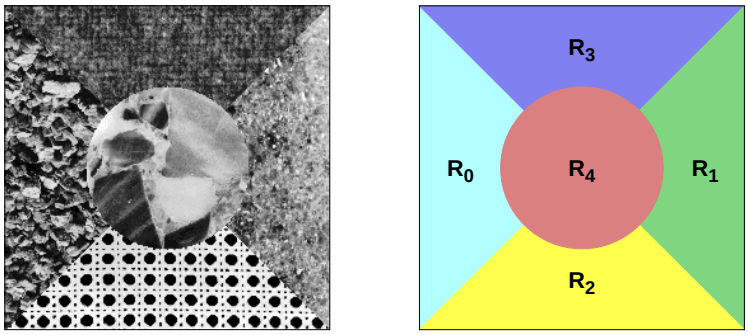
2. The pattern recognition system is limited to using one feature measurement for all classes. It is not possible to define a texture measure for some classes and a color measure for others.

We contrast pattern recognition with image segmentation, which does not require that the pattern classes are established *a priori*.

5.2.3 Image segmentation

Image segmentation is the process of partitioning an image $I[x, y]$ into non-overlapping segments R_i. A segmentation requires that the set of partitions $\{R_i\}$ is non-overlapping and complete as follows

$$\bigcup R_i = 1 \text{ and } \bigcap R_i = 0. \tag{5.1}$$



(a) (b)

Figure 5-3: Image segmentation is the process of segmenting the image into non-overlapping regions. (a) Brodatz texture composite image, (b) example results of image segmentation.

Image segmentation does not require that the pattern classes are known. In general, the objective is to separate the image into homogeneous regions [4]. For example, segmentation methods have been developed that perform fixed block segmentation using spatial quad-trees [155, 69] and region merging [111]. Other techniques, such as region growing [4], edge detection [81], texture segmentation [11, 67, 73, 86, 127, 176, 41, 24, 25, 138] and color segmentation [90, 122, 165, 14] have also been developed for segmenting images.

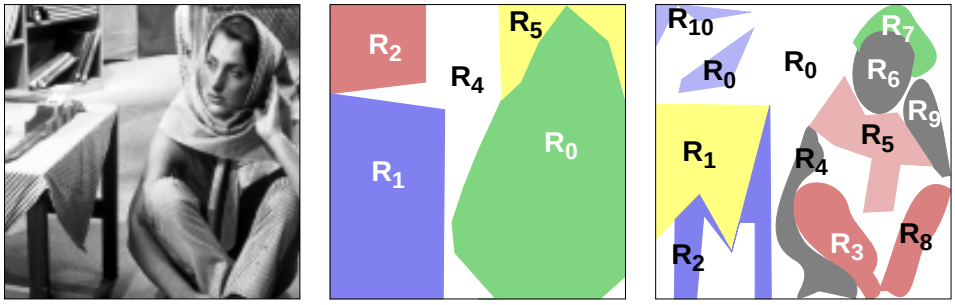
However, segmentation is not a well-defined problem in the case of unconstrained images. The pursuit of homogeneous regions is ill-posed for unconstrained images since the definition of the term "homogeneous" is application and domain-dependent. Furthermore, for unconstrained images there is no way of establishing the ground-truth segmentation in order to evaluate the correctness of any given segmentation $\{R_i\}$.

There are multiple ways to choose the partitions R_i to satisfy the constraints in Eq 5.1. For example, Figure 5-4 illustrates two ways to segment the Barbara image in order to satisfy Eq 5.1. In our application for extracting regions from unconstrained images, it is not possible to decide which is more acceptable. Therefore, we consider that both are acceptable by relaxing the constraints in Eq 5.1.

5.2.4 Region detection

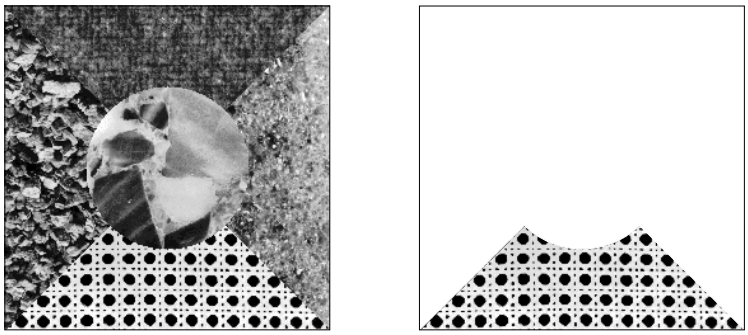
We define the problem of region detection as the determination of the existence and location of a model within an image. The problem is formulated as follows: given image $I[x, y]$ and model $\mathcal{L}_k$, detect the model in $I[x, y]$. The region detection problem is illustrated in Figure 5-5.

The problem of region detection is related to pattern recognition in that model patterns are used. However, the difference is that there is



(a) (b) (c)

Figure 5-4: Demonstration of multiple solutions for image segmentation (a) unsegmented image, (b) segmentation into six regions (b) segmentation into eleven regions.



(a) (b)

Figure 5-5: Region detection is the process of detecting a model pattern within an image. (a) Brodatz texture composite image, (b) example of the region detected for model pattern = "Cane."

1. no restriction on the number or collection Ω of models that are used. The complete set of models does not need to be defined *a priori*.

2. Each model may use a different feature measurement type v. Some model patterns are defined in terms of color and others in terms of texture.

When a library of models is used to extract regions of differing patterns from an image, the problem resembles that for image segmentation: a set of regions R_i are extracted from the image. However, the difference is that, here, $\bigcup R_i \leq 1$ and $\bigcap R_i \geq 0$. That is, the extracted regions may overlap and may not necessarily completely partition the image.

We now look more closely at the process of region extraction. We propose several HBP algorithms for extracting regions from images. In order to understand the HBP process, we present the framework of decision theory. We first show that the confidence HBP algorithm defined in [159] is related to the likelihood ratio test [163].

5.2.4.1 Decision theory framework

The simplest decision theory problem consists of several components: the source, the probabilistic transition mechanism, the observation space and the decision rule [163].

In our application, the first component corresponds to the source that generates the image. The source consists of the objects in the real-world and the light that is reflected from the objects into the camera. The probabilistic transition mechanism corresponds to the light pattern this is probabilistically generated by each object depicted in the image.

Given a model, we wish to determine the patterns in the image that correspond to the model. We formulate this as a simple two-hypothesis decision theory problem with the following two classes

$$\begin{aligned} H_1 &: \text{model present} \\ H_0 &: \text{model not present.} \end{aligned}$$

We assume that the probabilistic transition mechanism generates points according to the corresponding two conditional probability densities: points generated by the model follow $p_{\mathbf{r}|H_1}(\mathbf{R}|H_1)$ and other points follow $p_{\mathbf{r}|H_0}(\mathbf{R}|H_0)$.

Given the *a priori* probabilities for model presence P_1 and model absence P_0, and cost assignments for each course of action C_{ij} the expected cost of the risk $\Re$ is given by

$$\begin{aligned} \Re &= C_{11}P_1\mathbf{Pr}(\text{detect: } H_1|H_1) \\ &+ C_{01}P_1\mathbf{Pr}(\text{miss: } H_0|H_1) \\ &+ C_{10}P_0\mathbf{Pr}(\text{false alarm: } H_1|H_0) \\ &+ C_{00}P_0\mathbf{Pr}(\text{no detect: } H_1|H_1). \end{aligned} \tag{5.2}$$

By minimizing $\Re$ the likelihood ratio test is obtained by [163], choose H_1 when

$$\frac{p_{\mathbf{r}|H_1}(\mathbf{R}|H_1)}{p_{\mathbf{r}|H_0}(\mathbf{R}|H_0)} \overset{H_1}{>} \frac{P_0(C_{10}-C_{00})}{P_1(C_{01}-C_{11})}. \tag{5.3}$$

In our application, it is difficult to determine the *a priori* probabilities and cost assignments C_{ij}. We consider instead, the conditional probabilities for detection P_D and false alarm P_F, which are defined as follows

$$P_F = \int_{Z_1} p_{\mathbf{r}|H_0}(\mathbf{R}|H_0) \text{ and } P_D = \int_{Z_1} p_{\mathbf{r}|H_1}(\mathbf{R}|H_1). \tag{5.4}$$

By constraining P_F, the P_D can be maximized, which gives the Neyman-Pearson criterion,

$$\frac{p_{\mathbf{r}|H_1}(\mathbf{R}|H_1)}{p_{\mathbf{r}|H_0}(\mathbf{R}|H_0)} \overset{H_1}{>} \Lambda. \tag{5.5}$$

Here, Λ determines the tradeoff between P_F and P_D. We now apply this to the detection of regions, where the conditional probabilities are determined by the feature histograms.

5.2.4.2 Back-projection framework

In the case of images, the probabilistic transition mechanism generates an observation as each point in the image. We presented measures for these observations in terms of color in Chapter 2 and texture in Chapter 3. In this way, the features of regions are represented by histograms (color histograms and texture histograms). These histograms correspond directly to the transition mechanism probability distributions. That is, a model H_k generates a histogram $\mathbf{h}_k$ in accordance to $\mathbf{h}_k = p_{\mathbf{r},H_K}(\mathbf{R}|H_k)$, where $\mathbf{r}$ is the measurement vector.

In the general two-class detection problem, we define the histogram $\mathbf{h}_1$ for the model and $\mathbf{h}_0$ for non-model. From this, we reformulate the likelihood detection using the feature histograms as follows

$$\frac{\mathbf{h}_1}{\mathbf{h}_0} \overset{H_1}{>} \Lambda. \tag{5.6}$$

We now present the HBP algorithms, which are variations of Eq 5.6.

5.3. Back-projection algorithms

The HBP process is carried out using the following histograms:

1. model histogram, $\mathbf{h}_q = \mathbf{h}_1$,
2. target image histogram, $\mathbf{h}_t = \mathbf{h}_0$, and
3. local region histogram, $\mathbf{h}_l$.

The back-projection systems slide a window through the target image and compute the likelihood that each image point belongs to the model. The likelihood image is then filtered to reduce noise. By thresholding at Λ, the locations of the model are determined in the image.

We investigate five formulations of back-projection: (1) confidence back-projection, (2) quadratic confidence back-projection, (3) local histogramming, (4) binary set back-projection and (5) single-element back-projection. The back-projection algorithms consist of several stages: back-projection, smoothing, thresholding and detection.

5.3.1 Confidence back-projection ($\mathcal{BP}1$)

Confidence back-projection was proposed by Swain and Ballard [159]. The confidence histogram $\mathbf{h}_r$ is first generated by taking the ratio of the model $\mathbf{h}_q$ and target histograms $\mathbf{h}_t$

$$h_r[m] = \frac{h_q[m]}{h_t[m]}. \tag{5.7}$$

Each point in the image is assigned a likelihood as follows

$$\text{let } \; k = I[x,y], \quad \text{ then } \quad I'[x,y] = h_r[k]. \tag{5.8}$$

The image $I'[x,y]$ is then smoothed to reduce noise giving $I''[x,y] = B[x,y] * I'[x,y]$, where $*$ gives the 2-D convolution with blurring filter $B[x,y]$ as follows

$$I''[x,y] = \sum_{w=0}^{W-1} \sum_{h=0}^{H-1} B[x-w, y-h] I'[w,h]. \tag{5.9}$$

Finally, $I''[x,y]$ is thresholded. The peaks reveal the locations of the detected model $\mathbf{h}_q$. One drawback of $\mathcal{BP}1$ is that the back-projection only compares like-bins. This is similar to the drawback of using the Minkowski-form metrics for computing histogram distance (see Section 4.3.1). To address this, we develop $\mathcal{BP}2$ to consider the similarity of each target element $h_t[m]$ to all the elements in the model histogram $\mathbf{h}_q$.

5.3.2 Quadratic confidence back-projection ($\mathcal{BP}2$)

In quadratic confidence histogram back-projection, $\mathbf{h}_r$ is generated by taking the ratio of the model $\mathbf{h}_q$ and target histograms $\mathbf{h}_t$ as follows

$$h_r[m] = \frac{\sum_n h_q[m] A_{m,n}}{h_t[m]}, \tag{5.10}$$

where $A_{m,n}$ defines the similarity of the histogram elements indexed by m and n. Each point in the image is assigned a likelihood as follows

$$\text{let } \; k = I[x,y], \quad \text{ then } \quad I'[x,y] = h_r[k]. \tag{5.11}$$

The image $I'[x,y]$ is then smoothed to reduce noise and thresholded to reveal the most likely locations of $\mathbf{h}_q$.

5.3.3 Local histogramming ($\mathcal{BP}3$)

In local histogramming, the smoothing operation and back-projection steps are combined. In this case, for each point in the target image $I[x,y]$, a local neighborhood histogram $\mathbf{h}_l$ is computed and its distance to the model histogram $\mathbf{h}_q$ determines whether each image point belongs to the model. Define $\mathbf{h}_l^{xy}$ to be the local neighborhood histogram for the region R_{xy}^{wh} in image $I[x,y]$, centered at point (x,y) with size $w \times h$. Then, let

$$I''[x,y] = d_{q,l}(\mathbf{h}_q, \mathbf{h}_l^{xy}). \tag{5.12}$$

$\mathcal{BP}3$ is more robust than $\mathcal{BP}1$ and $\mathcal{BP}2$ since the most effective distance metric $d_{q,l}$ may be used (see Chapter 4).

5.3.4 Binary set back-projection ($\mathcal{BP}4$)

In binary set back-projection, the target image points are assigned 1 or 0 depending on membership in the model binary set $\mathbf{s}_q$ where $\mathbf{s}_q \in \mathcal{B}^M$. In this way, $\mathcal{BP}4$ is extremely simple as follows

$$\text{let } \; k = I[x,y], \; \text{ then } \; I'[x,y] = s_q[k]. \tag{5.13}$$

The image $I'[x,y]$ is then smoothed to reduce noise and thresholded to reveal the most likely locations of $\mathbf{s}_q$.

5.3.5 Single element quadratic back-projection ($\mathcal{BP}5$)

Single element quadratic back-projection combines $\mathcal{BP}2$ and $\mathcal{BP}4$. The target image points are weighted by similarity to the back-projection element m, where $s_q[m] = 1$ and $s_q[k] = 0, \forall_{k \neq m}$ such that

$$\text{let } \; k = I[x,y], \; \text{ then } \; I'[x,y] = A_{m,k}. \tag{5.14}$$

The image $I'[x,y]$ is then smoothed to reduce noise and thresholded to reveal the most likely locations of $\mathbf{s}_q$.

5.4. Back-projection region extraction examples

We demonstrate the back-projection methods by detecting models of color and texture in example images. The results are given in Figures 5-6, 5-7, 5-8 and 5-9.

5.4.1 Color histogram back-projection

Figure 5-6 illustrates the results of color HBP using algorithms ($\mathcal{BP}1 \ldots \mathcal{BP}4$). The model image depicts a sample of "yellow grass." The first row depicts the back-projection images. The second row depicts the filtered images. We see that $\mathcal{BP}2$ improves the detection results over $\mathcal{BP}1$, however, $\mathcal{BP}3$ best determines the location of the model. The binary set back-projection $\mathcal{BP}4$ detects the model sufficiently well, which indicates that $\mathcal{BP}4$ has a role as a pre-filter before a more expensive back-projection, i.e, $\mathcal{BP}3$.

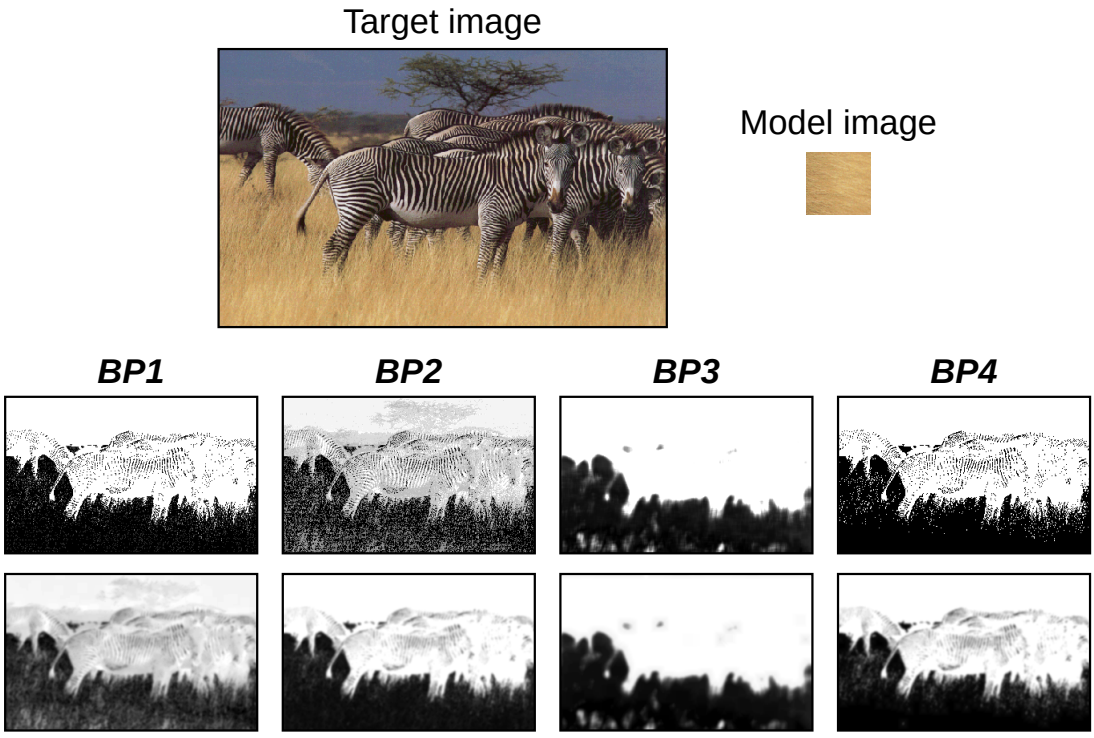


Figure 5-6: Color histogram back-projection comparing methods $\mathcal{BP}1$, $\mathcal{BP}2$, $\mathcal{BP}3$, and $\mathcal{BP}4$ in detection of yellow grass.

5.4.2 Object detection and histogram back-projection

Figure 5-7 illustrates the results of detecting an object within a crowded image using color HBP. The model depicts the "Waldo" character. The objective is to "Find Waldo" within the crowded scene. This problem was motivated by Ennesser and Medioni and was addressed in [43]. The first row depicts the back-projection images. The second row depicts the filtered images. The "Find Waldo" problem is difficult because many false Waldo's are present. In this difficult detection problem, $\mathcal{BP}3$ clearly gives the best performance. The others detect the location of Waldo but have several false alarms in addition.

5.4.3 Texture histogram back-projection

Figure 5-8 illustrates the results of texture HBP using algorithms ($\mathcal{BP}1 \ldots \mathcal{BP}5$). The wavelet images for the target and model images are also depicted. The first row depicts the back-projection images. The second row depicts the filtered images. The model image depicts a set of vertical zebra stripes. The objective is to detect the zebra stripes within the image. We see that all back-projection methods detect the stripes well. However, again $\mathcal{BP}3$ best determines the location of the model. The binary set back-projection $\mathcal{BP}4$ detects the model sufficiently well for texture back-projection.

5.4.4 Combining color and texture histogram back-projection

We combine color and texture back-projections results to better detect regions. Figure 5-9 illustrates an example of combining color and texture back-projection. Neither color or texture alone sufficiently extracts the flower region from the image. However, by combining the individual results, the model is better detected.

5.5. Region extraction systems

We now present systems for automatically generating the VFL's and extracting regions from the images.

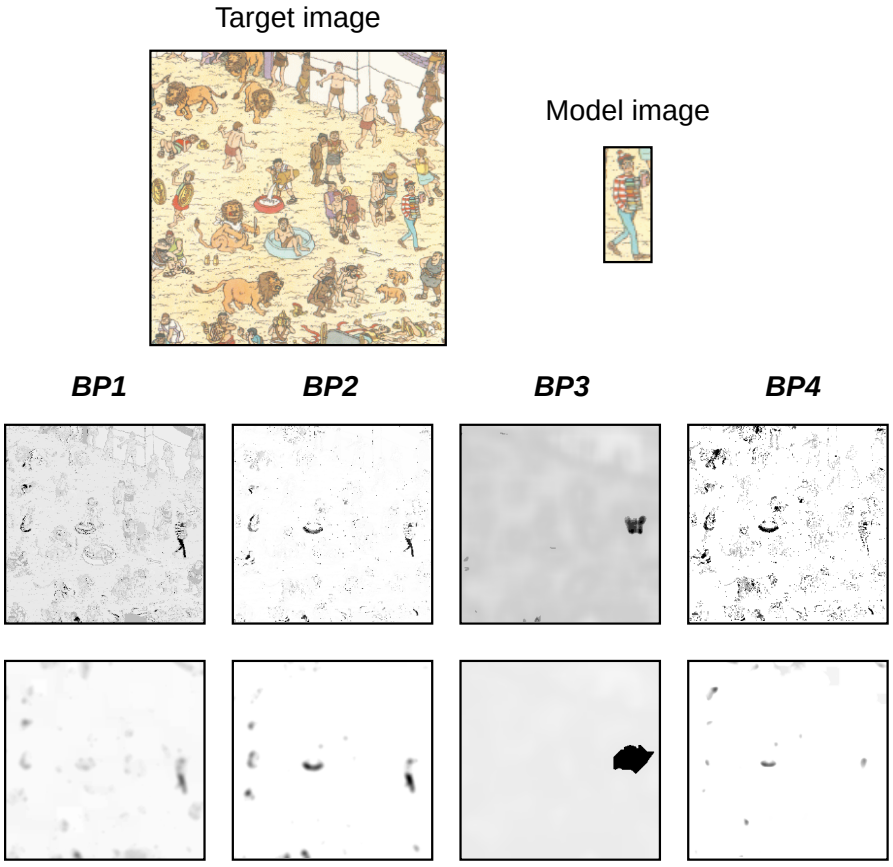


Figure 5-7: Color histogram back-projection comparing methods $\mathcal{BP}1$, $\mathcal{BP}2$, $\mathcal{BP}3$, and $\mathcal{BP}4$ in "Find Waldo".

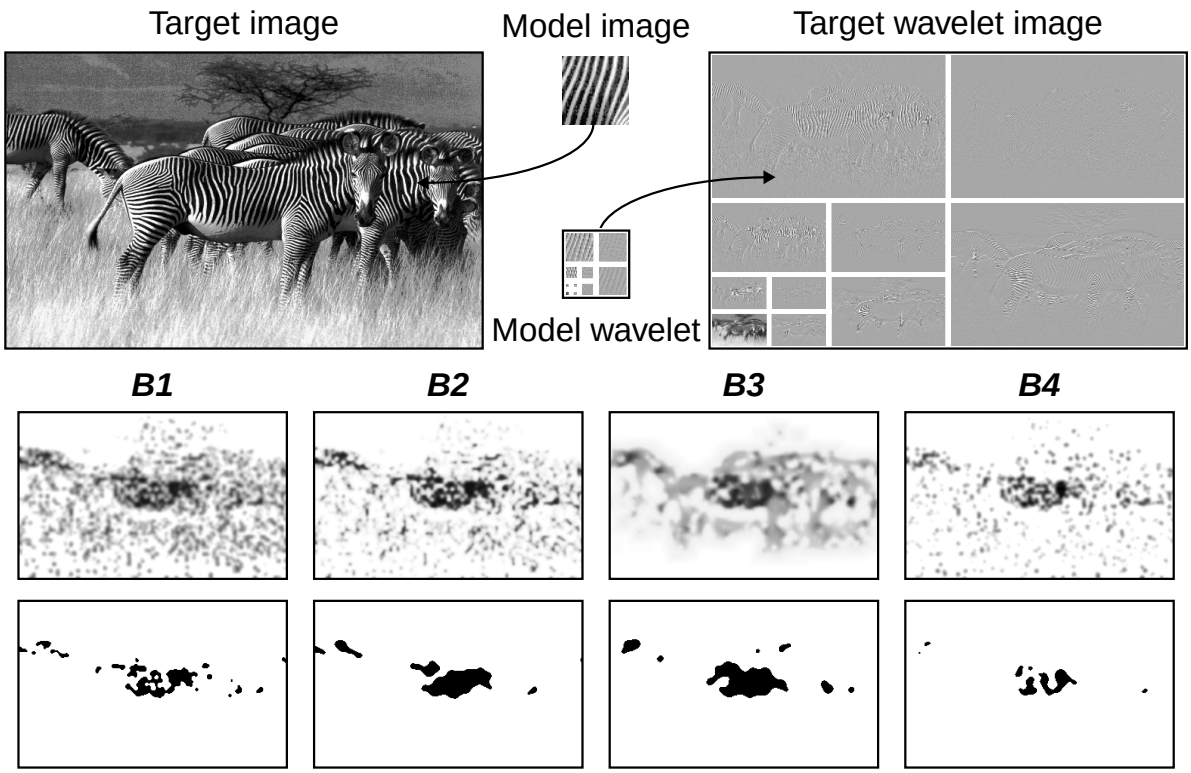


Figure 5-8: Texture histogram back-projection comparing methods $\mathcal{BP}1$, $\mathcal{BP}2$, $\mathcal{BP}3$, and $\mathcal{BP}4$ in detection of zebra stripes.

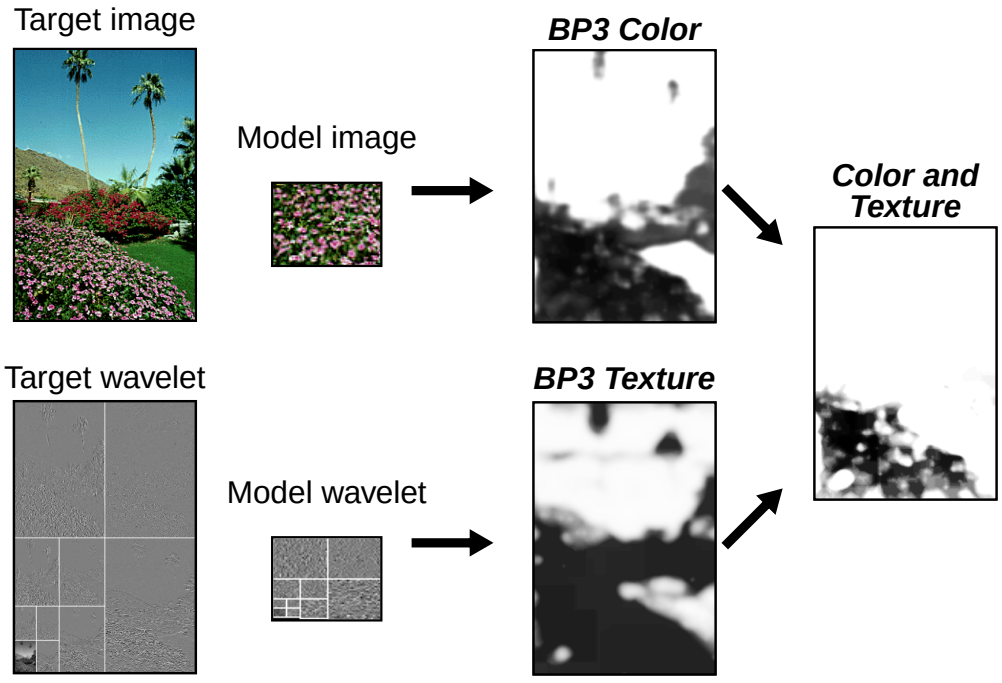


Figure 5-9: Combined color/texture back-projection.

5.5.1 Single-element quadratic back-projection system

The single-element back-projection algorithm $\mathcal{BP}5$ is a special case in that it is suited for detecting only models with single-element features. We develop a simple process for single-element region extraction by iterating over single-element histograms. This provides a novel approach for automated region extraction. In this case, the VFL is generated automatically and adaptively to each image.

For example, we generate first, for each image, a histogram $\mathbf{h}_t$ (color or texture). For each m such that $h_t[m] \geq \tau$ we generate a binary set such that $s^m[m] = 1$ and $\forall_{i \neq m} s^m[i] = 0$. The collection $\{\mathbf{s^m}\}$ of binary sets constructs a VFL for the image.

We then back-project, using $\mathcal{BP}5$, each binary set $\mathbf{s^m}$ onto the image. The detected regions are extracted and are made available for integrated spatial and feature querying. The results of two examples of single-color quadratic back-projection system are illustrated in Figure 5-10 and Figure 5-11.

We consider a generalization of the single-element back-projection system in which single-, and multiple-element binary sets are back-projected using $\mathcal{BP}5$. In general, we develop a binary set iteration procedure for generating the VFL adaptively to each image.

5.5.2 Binary set iteration (BSI) system

In the binary set iteration (BSI) and back-projection system, two histograms are created: the target image histogram $\mathbf{h}_t$ and a residue histogram $\mathbf{h}_r$. We utilize two thresholds for each bin, τ_t and τ_r, in the iteration process as follows: for each m such that $h_t[m] \geq \tau_t$ the single-element back-projection procedure is performed. Then, the histograms for all the extracted regions are summed to generate $\mathbf{h}^{(1)}$. We compute $\mathbf{h}_r = \mathbf{h}_t - \mathbf{h}^{(1)}$. Then, for all pairs of elements (m, n) where $h_r[m] \geq \tau_r$ and $h_r[n] \geq \tau_r$, binary color sets are generated such that $s^{(m,n)}[m] = 1$ and $s^{(m,n)}[n] = 1$ and $\forall_{i \neq m,n} s^m[i] = 0$. These two-element binary sets are back-projected using $\mathcal{BP}2$. The process is repeated until $\forall_m h_r[m] \leq \tau_r$.

5.6. Summary

We presented a strategy for extracting regions from images in an image collection. By using a visual feature library (VFL) and the histogram back-projection (HBP) system, the feature elements in the VFL are tested-in and extracted-from the images. We presented two systems for generating the VFL's automatically using binary sets.

We also presented and compared five algorithms for histogram back-projection and demonstrated their performance in detecting regions from images. In the color and texture back-projection examples, the back-projection technique based upon local histogramming ($\mathcal{BP}3$) performs best. However, computationally cheaper methods such as binary set back-projection ($\mathcal{BP}4$) perform sufficiently well in detecting regions in the example back-projections.

We use the back-projection system and the VFL to develop the overall strategy for the integrated spatial and feature query system. By using the representations for color and texture presented in Chapters 2 and 3, the feature similarity measures of Chapter 4 and the region extraction techniques presented here, we have the tools for developing the complete spatial and feature query system. Before presenting the query process, we describe next in Chapter 6 a new method for adaptive image analysis and compression. The compression of images is a key component for storage and transmission in the image systems.

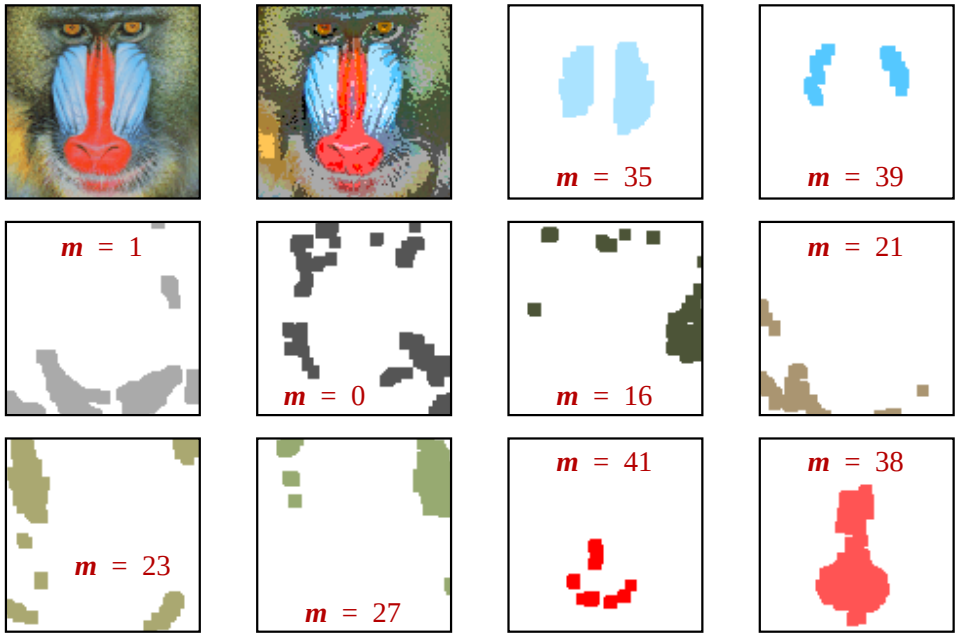


Figure 5-10: Results of the single color quadratic back-projection system ($\mathcal{BP}5$) for automated region extraction from the "Baboon" image.

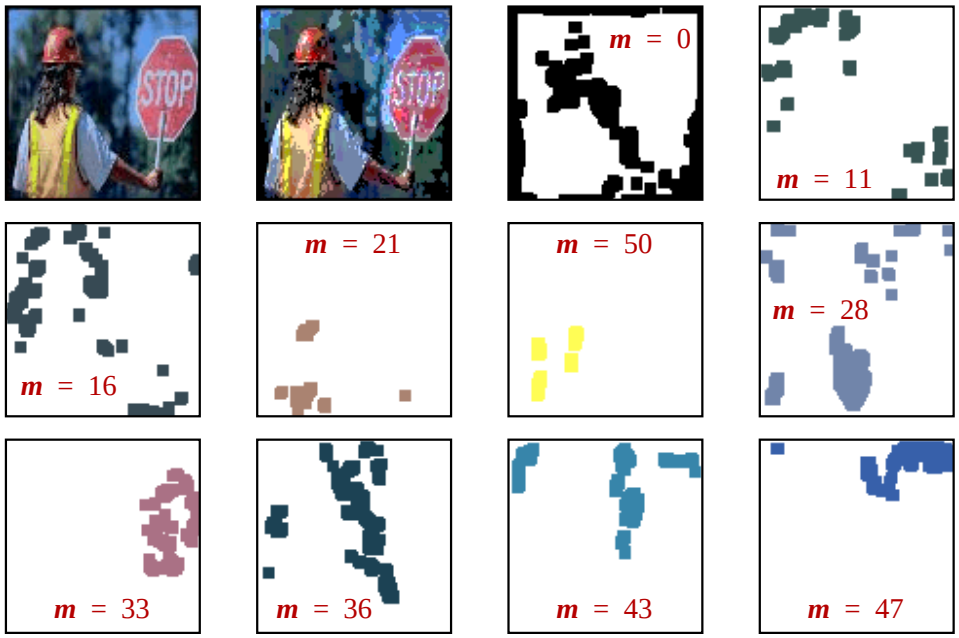


Figure 5-11: Results of the single color quadratic back-projection system ($\mathcal{BP}5$) for automated region extraction from the "Stop sign" image.

Chapter 6

Adaptive Image Analysis and Compression

6.1. Introduction

The transformation of signals is of critical importance in applications of compression, feature extraction and noise reduction. It provides a re-organization of data by which the signal can be better analyzed, prioritized, quantized and/or discarded. Recently, algorithms have been proposed, such as those based upon, wavelet packets [31, 124], quadtree segmentation [136, 132], matching pursuits [94, 6], and the double-tree [62], for generating bases adaptively to the signal. However, each of these has shortcomings. Wavelet packets do not adapt to local spatial content. The quadtree provides poor energy compaction. The double-tree provides an asymmetric adaptation in the space and frequency domains. Matching pursuits compute a sub-optimal expansion.

In this paper, we present a joint adaptive space and frequency best basis selection method. The joint adaptive space and frequency (JASF) graph provides a complete and symmetric expansion of an image into a library of spatially and frequency localized basis elements. By implementing the frequency expansion in a partitionable-form, any near-redundant terms are prevented, allowing the basis elements to be indexed by the JASF graph.

The JASF graph provides for (1) the efficient expansion of the image into a library of basis elements, (2) detection of the best basis and (3) coding of the best basis. The JASF graph examines exponentially more bases than previous methods. For example, for a JASF expansion of depth = 6, the JASF graph examines 10^{20} times as many bases as a double-tree of the same size.

6.1.1 Signal expansion

Signal expansions are produced by linear transforms and/or filter banks. Fast algorithms make them viable for practical systems. For example, they are well suited to applications such as real-time video coding, where both the encoder and decoder need to have low complexity. The constraints allow for little adaptivity in the signal expansion. As a consequence, fixed transforms such as the discrete cosine transform (DCT) have become most common for image compression (JPEG) [168] and video compression (MPEG) [49].

However, there are many new applications that support an asymmetric coding model whereby the constraints at the encoder and the decoder differ. For example, in digital image and video library applications the data is typically encoded once – off-line, and is stored. Real-time compression is not required and the encoder does not need to be of low complexity. The compressed data is retrieved later for purposes of analysis [139], decompression and viewing. The encoded images and videos need to be quickly and cheaply decompressed and possibly analyzed directly in the compressed domain [17].

In general, digital and image libraries require new and efficient compression systems that jointly (1) decrease the code size, (2) lower the visible distortion, and (3) improve access to visual content and image features [115, 18]. Given these new applications, it is worthwhile to investigate new procedures for the adaptive compression of images in the design of the image systems.

6.1.2 Adaptive signal expansions

Recent methods in image compression adaptively decompose the images in order to compact the information. The purpose is to derive a particular segmentation, transformation or filter bank that is customized to each image. In several solutions, tree- and graph- structured expansions are used to generate libraries of basis elements [31, 124, 170, 62, 140, 144].

In these methods, a *basis element* is defined to consist of one set *basis functions* that is generated and coded as a group; each basis element corresponds to one node in the tree or graph. The objective of the compression algorithm is to identify the best basis which consists of a "complete" set of basis elements that have the least total coding cost. The completeness constraint enables perfect reconstruction (PR) in the absence of quantization. The various tree- and graph-expansions differ in the sizes of their basis element libraries and/or the number of unique bases they provide.

The two main approaches decompose the images either by frequency, such as wavelet packets (WP), or spatially, such as quadtree (QT) segmentation. Hybrid approaches such as the double-tree (DT) algorithm incorporate both segmentation and frequency expansion. However, the symmetric joint space and frequency expansion of images has not been addressed.

For example, Figure 6-1 illustrates the set of (18) possible time-segmentations and frequency expansions for a four point 1-D signal. Notice that strategies based upon adaptive segmentation-only or frequency-only generate only a fraction of the available bases, only five bases each. The DT generates a greater number of bases (14), however, because of the DT's asymmetric treatment of segmentation and frequency expansion, the DT still misses four bases. The JASF graph generates the complete set of (18) bases. We demonstrate that in compressing images, the extra bases generated by the JASF graph greatly improve the potential compression. We describe these four methods in more detail.

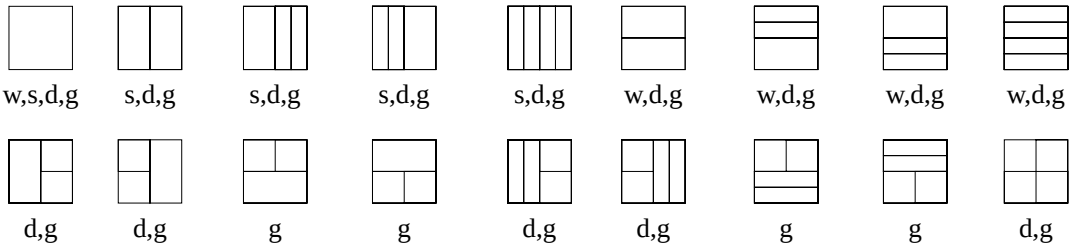
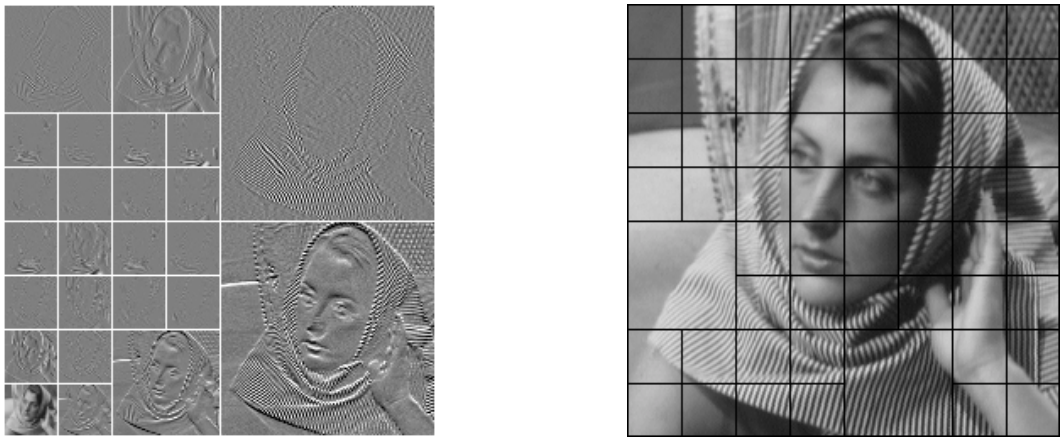


Figure 6-1: Example time-frequency bases for a four point 1-D signal. The squares depict the 18 possible ways to partition the time-frequency plane. The vertical lines correspond to segmentations in time while the horizontal lines correspond to segmentations in frequency: s = segmentation-only, w = wavelet packet (WP)-tree, d = double-tree (DT), g = JASF graph.

Wavelet packet (WP)-tree An algorithm for adaptive selection of the best frequency, or wavelet packet basis was proposed by Coifman, Quake, Meyer and Wickerhauser [31]. The wavelet packet (WP) algorithm generates a library of orthonormal functions that are derived from a single filter kernel. The WP algorithm searches through the basis element library to find the best basis for the image [124].

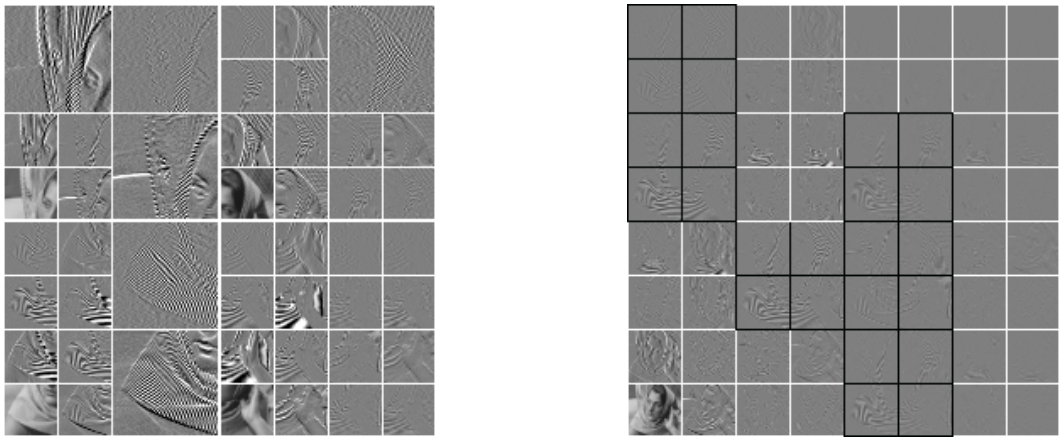
The WP expansion is generated using a tree-structured cascade of filtering and downsampling operations (filter banks). The tree also guides the search for best basis [31, 170, 124]. But, an important drawback of the WP-tree is that the expansion is performed on the entire image. For example, a WP-tree expansion is illustrated in Figure 6-2(a). The WP-tree cannot adapt to variations in spatial content in separate regions of the image – or to non-stationarity.

Spatial quadtrees (QTs) Spatial quadtrees (QT) generate a hierarchy of spatial segmentations of the image. Figure 6-2(b) depicts an example of QT segmentation. Since the spatial QT can adapt to the different spatial regions, it better adapts to image non-stationarity than the WP-tree. However, the spatial QT does not have the same capacity as the WP-tree for compacting the energy of the image into a small set of most significant elements.



(a) (b)

Figure 6-2: (a) Wavelet packet (WP)-tree adapts to global frequency, (b) spatial quadtree (QT) adapts to spatial content. (White lines = frequency expansion, black lines = spatial segmentation)



(a) (b)

Figure 6-3: (a) Double-tree (DT) expansion treats space and frequency asymmetrically, (b) JASF graph expansion treats space and frequency symmetrically. (White lines = frequency expansion, black lines = spatial segmentation)

Double-tree (DT) To address the problem of non-stationarity without sacrificing energy compaction, the double-tree (DT) algorithm was proposed by Herley, Kovačević, Ramchandran and Vetterli [62]. The DT finds the best WP basis for a hierarchy of binary segmentations of the signal. The overall best basis search identifies the best overall segmentation and best WP basis for the segments.

However, the DT does not exploit the full potential of a joint space and frequency expansion. The insufficiency results from the asymmetric treatment of the space and frequency operations in the tree cascades, as illustrated in the example of a DT expansion of the Barbara image in Figure 6-3(a). Furthermore, we will show that a "single critical pruning decision" reduces the DT to a WP-tree. We have observed for images that the DT is most typically pruned into a WP-tree. Therefore, in practice, the DT offers no advantage over the WP-tree.

JASF graph The JASF algorithm treats the space and frequency operations symmetrically in a graph structured cascade [140, 144]. An example of a JASF graph expansion of the Barbara image is illustrated in Figure 6-3(b). Both the DT and its dual are embedded in the JASF graph. The JASF graph requires that the basis functions are modified to generate a partitionable frequency expansion. While this does impact the frequency expansion, it allows the JASF graph to examine a greater number of bases, and to adapt to the image in both space and frequency.

6.1.3 Outline

In this paper, we present the JASF graph image expansion and basis selection method. In Section 6.2., we first review the framework for signal expansion using filter banks. We also develop the theory of "partitionable" frequency expansions which are fundamental to the JASF graph. In Section 6.3., we present the process of spatial segmentation of images using spatial QTs. In Section 6.4., we integrate frequency expansion and segmentation by cascading the "partitionable" filter banks and spatial QT segmentations. We show that by using "partitionable" expansions, the generation of the basis library and the reconstruction of the image from the library elements follows a number of equivalent paths.

In Section 6.5., we show that the JASF graph greatly increases the available number of bases. In Section 6.6., we describe the basis selection procedure which consists of selecting elements from the JASF library via the graph-structure [150]. In Section 6.5.1, we compare the sizes of the various tree- and graph-structured basis element libraries and number of bases they provide. Finally, in Section 6.7. we demonstrate improved compression performance using the JASF graph over the WP-tree spatial QT and DT methods.

6.2. Image signal expansion

We first consider a 1-D signal, the extension of the 1-D signal expansion to 2-D discrete-space images is straightforward and will be explained when necessary. We first consider that a discrete-time filter bank produces a series expansion of a 1-D signal into a set of basis functions.

6.2.1 Series expansion

A series expansion represents the signal using a set of orthonormal basis functions $\{\phi_k\}$ such that

$$x[n] = \sum_{k \in \mathcal{Z}} \langle \phi_k, x \rangle \phi_k[n], \quad n \in \mathcal{Z}. \tag{6.1}$$

Here $\langle \phi_k[n], \phi_l[n] \rangle = \delta[k-l]$ dictates an orthonormality constraint and the inner product is given by $\langle x, y \rangle = \sum_{n=-\infty}^{\infty} x^*[n] y[n]$. The expansion of x into the set of basis functions $\{\phi_k\}$ produces the transform coefficients X where $X[k] = \langle \phi_k, x \rangle$.

6.2.2 Filter bank

A two-channel filter bank implements an orthonormal expansion when even shifts of the analysis h_i and synthesis g_i filters are related to the basis functions by [167]

$$\begin{aligned} h_0[2k-n] &= g_0[n-2k] = \phi_{2k}[n], \text{ and} \\ h_1[2k-n] &= g_1[n-2k] = \phi_{2k+1}[n]. \end{aligned} \tag{6.2}$$

In the filter bank (FB), the inner products for the signal expansion are accomplished using convolutions with time-shifted, time-reversed versions of a finite set of basis function prototypes. Furthermore, the two FB channels (y_0, y_1) correspond to the odd $X[2k+1]$ and even $X[2k]$ transform coefficients

$$y_0[k] = X[2k] \quad \text{and} \quad y_1[k] = X[2k+1]. \tag{6.3}$$

The subbands y_0 and y_1 are generated by the FB by the analysis frequency expansion matrices $\mathbf{H}_0$ and $\mathbf{H}_1$, respectively, as follows, for $i = \{0, 1\}$,

$$y_i = \mathbf{H}_i[\ldots\ x[0], x[1]\ \ldots\ x[N-1]\ \ldots]^T \tag{6.4}$$

By using finite impulse response (FIR) filters of length L, the frequency expansion matrices $\mathbf{H}_i$ are constructed from the filters h_0 and h_1 as follows, for $i = \{0, 1\}$,

$$\mathbf{H}_i = \begin{pmatrix} \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ h_i[L-1] & h_i[L-2] & \cdots & h_i[0] & 0 & \cdots & 0 & 0 \\ 0 & 0 & h_i[L-1] & h_i[L-2] & \cdots & h_i[0] & 0 & 0 \\ \ddots & \ddots & & \ddots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & h_i[L-1] & h_i[L-2] & \cdots & h_i[0] \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \end{pmatrix}. \tag{6.5}$$

Following from Eq. 6.1 and Eq. 6.2, and by similarly defining $\mathbf{G}_i$ from the $g_i[m]$'s in Eq 6.2, the signal is resynthesized by $\hat{x} = (\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1)x$. The analysis frequency-expansion is orthonormal when $\mathbf{G}_i = \mathbf{H}_i^T$ and the FB satisfies the following condition of perfect reconstruction (PR) [167], where $\mathbf{I}$ is the identity matrix,

$$\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1 = \mathbf{I}. \tag{6.6}$$

The two-channel FB is illustrated in Figure 6-4. In the z-domain, the output of the analysis section are the two subbands, $Y_i(z)$, where $i = 0, 1$, which are given by,

$$\begin{pmatrix} Y_0(z^2) \\ Y_1(z^2) \end{pmatrix} = \frac{1}{2} \begin{pmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{pmatrix} \begin{pmatrix} X(z) \\ X(-z) \end{pmatrix}. \tag{6.7}$$

The signal $\tilde{X}(z)$ is reconstructed from subbands $Y_0(z)$ and $Y_1(z)$ by the synthesis filters G_i as follows

$$\tilde{X}(z) = G_0(z)Y_0(z^2) + G_1(z)Y_1(z^2). \tag{6.8}$$

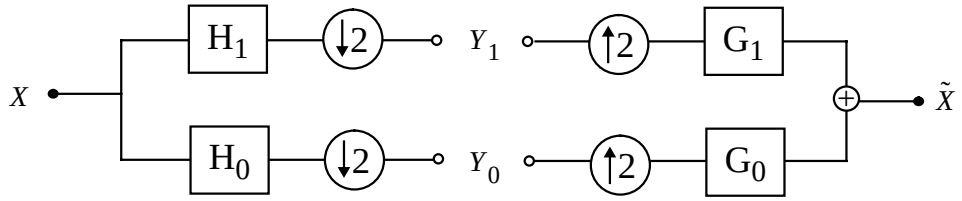


Figure 6-4: Two-channel filter bank (FB). Analysis filtering with $\mathbf{H}_i$ and subsampling generate subbands Y_0 and Y_1. Upsampling and synthesis filtering with $\mathbf{G}_i$ and adding reconstruct $\tilde{X} = X$.

6.2.3 Finite signal expansion

Since images are finite signals, say size $N \times N$, in order to maintain the PR property of the FB, the frequency-expansion matrices $\mathbf{H}_0$ and $\mathbf{H}_1$ must be modified to generate a finite set of basis functions. This is achieved by utilizing circular convolution (or equivalently, periodic signal extension) in the filtering process. The modified filters h_i and g_i are periodically related to the basis functions by

$$\begin{aligned} h_0[2k-n] &= g_0[n-2k] = \phi_{2k}[n \bmod N], \\ h_1[2k-n] &= g_1[n-2k] = \phi_{2k+1}[n \bmod N]. \end{aligned} \tag{6.9}$$

The PR condition is retained for the expansion of a finite signal since $\mathbf{G}_i = \mathbf{H}_i^T$ and $\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1 = \mathbf{I}$.

For example, for a 1-D time series of length $N = 8$, and filters of length $M = 4$, the subbands y_0 and y_1

are generated by

$$\underbrace{\begin{pmatrix} y_i[0] \\ y_i[1] \\ y_i[2] \\ y_i[3] \end{pmatrix}}_{\mathbf{y}_i} = \underbrace{\begin{pmatrix} h_i[1] & h_i[0] & 0 & 0 & 0 & 0 & h_i[3] & h_i[2] \\ h_i[3] & h_i[2] & h_i[1] & h_i[0] & 0 & 0 & 0 & 0 \\ 0 & 0 & h_i[3] & h_i[2] & h_i[1] & h_i[0] & 0 & 0 \\ 0 & 0 & 0 & 0 & h_i[3] & h_i[2] & h_i[1] & h_i[0] \end{pmatrix}}_{\mathbf{H}_i} \underbrace{\begin{pmatrix} x[0] \\ x[1] \\ \vdots \\ x[7] \end{pmatrix}}_{\mathbf{x}}. \tag{6.10}$$

The resynthesis of $x = \hat{x}$ is provided by $\hat{x} = \sum_{i=0}^{1} \mathbf{G}_i y_i = \sum_{i=0}^{1} \mathbf{G}_i \mathbf{H}_i x_i$, which gives

$$\begin{pmatrix} \hat{x}[0] \\ \hat{x}[1] \\ \vdots \\ \hat{x}[7] \end{pmatrix} = \sum_{i=0}^{1} \begin{pmatrix} h_i[1] & h_i[3] & 0 & 0 \\ h_i[0] & h_i[2] & 0 & 0 \\ 0 & h_i[1] & h_i[3] & 0 \\ 0 & h_i[0] & h_i[2] & 0 \\ 0 & 0 & h_i[1] & h_i[3] \\ 0 & 0 & h_i[0] & h_i[2] \\ h_i[3] & 0 & 0 & h_i[1] \\ h_i[2] & 0 & 0 & h_i[0] \end{pmatrix} \begin{pmatrix} y_i[0] \\ y_i[1] \\ y_i[2] \\ y_i[3] \end{pmatrix}. \tag{6.11}$$

We consider in addition to constraining the basis function set to be finite, we further constrain the frequency-expansion matrices to be "partitionable." The partitionable frequency expansion generates, simultaneously, an orthonormal expansions of the full signal and of the segments of the signal.

6.2.4 Partitionable expansions

The orthonormal "partitionable" frequency expansion is constructed by concatenating sub-expansions which are orthonormal over segments of the signal. In general, an orthonormal expansion produced by filter banks is not necessarily partitionable, but it may be made partitionable as we explain shortly. Frequency analysis expansions $\mathbf{H}_0$ and $\mathbf{H}_1$ that are at least M-partitionable are required in order to produce the JASF graph expansion of depth M.

6.2.4.1 1-partitionable expansion

We begin by first defining a 1-partitionable frequency expansion (1-PFE) as follows,

Definition 1 *The frequency expansion matrix* $\mathbf{H}$ *is* 1*-partitionable if it has only zeros in the upper right and lower left quadrants.*

If the frequency expansion matrix set $\{\mathbf{H}_0, \mathbf{H}_1\}$ generates a 1-PFE then the overall expansion consists of two independent expansions over the two half-length signals, as we now explain. First, observe that for QMF filter banks the frequency expansion matrices $\mathbf{H}_i$, $i \in \{0, 1\}$ can be written in the following form [167]

$$\mathbf{H}_i = \begin{pmatrix} \mathcal{H}_i^a & \mathcal{H}_i^b \\ \mathcal{H}_i^b & \mathcal{H}_i^a \end{pmatrix}. \tag{6.12}$$

We construct the 1-PFE from $\mathbf{H}_i$ by $\mathcal{H}_i = \mathcal{H}_i^a + \mathcal{H}_i^b$ as follows:

$$\mathbf{H}_i^{(1)} = \begin{pmatrix} \mathcal{H}_i & \mathbf{0} \\ \mathbf{0} & \mathcal{H}_i \end{pmatrix} = \begin{pmatrix} \mathcal{H}_i^a + \mathcal{H}_i^b & \mathbf{0} \\ \mathbf{0} & \mathcal{H}_i^a + \mathcal{H}_i^b \end{pmatrix}. \tag{6.13}$$

We now state the following useful results and definition: if the original frequency expansion set $\{\mathbf{H}_0, \mathbf{H}_1\}$ satisfies the condition of perfect reconstruction and orthonormality, that is $\mathbf{H}_0^t \mathbf{H}_0 + \mathbf{H}_1^t \mathbf{H}_1 = \mathbf{I}^N$, then the 1-PFE set $\{\mathbf{H}_0^{(1)}, \mathbf{H}_1^{(1)}\}$ is orthonormal and the set $\{\mathcal{H}_0, \mathcal{H}_1\}$ also generates an orthonormal expansion of length $N/2$.

Definition 2 *The frequency expansion (1-PFE) matrix set* $\{\mathbf{H}_0^{(1)}, \mathbf{H}_1^{(1)}\}$ *is* 1*-partitionable and orthonormal if* $\{\mathbf{H}_0, \mathbf{H}_1\}$ *satisfies the perfect reconstruction condition. For proof, see A..*

It is also obvious to see that for a FB using length $L = 2$ filters, such as the Haar expansion, the $\mathbf{H}_i$ generate a 1-PFE since, $\mathcal{H}_i = \mathcal{H}_i^a$ and $\mathcal{H}_i^b = 0$, which gives

$$\mathbf{H}_i^{(1)} = \begin{pmatrix} \mathcal{H}_i & \mathbf{0} \\ \mathbf{0} & \mathcal{H}_i \end{pmatrix} = \left(\begin{array}{cccc|cccc} h_i[1] & h_i[0] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_i[1] & h_i[0] & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & h_i[1] & h_i[0] & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_i[1] & h_i[0] \end{array}\right). \tag{6.14}$$

6.2.4.2 1-partitionable convolution

The frequency expansion matrices $\mathbf{H}_i$ for $i \in \{0, 1\}$ can be made 1-partitionable by summing the quadrants as illustrated in Eq 6.13. This is equivalent to wrapping the filters at the $N/2$ boundary (using circular convolutions with period $= N/2$) as long as the filter length $L \leq N/2$. This is implemented by first segmenting the length N signal into two half length $N/2$ signals and expanding each separately. In the 1-PFE, the modified filters $h_i^{(1)}$ and $g_i^{(1)}$ are periodically related to the basis functions by

$$\begin{aligned} h_0^{(1)}[2k-n] = \quad g_0^{(1)}[n-2k] = & \begin{cases} \phi_{2k}[n \bmod N/2] & (2k < N/2 \text{ and } n < N/2) \text{ or} \\ & (2k \geq N/2 \text{ and } n \geq N/2) \\ 0 & \text{otherwise} \end{cases} \\ h_1^{(1)}[2k-n] = \quad g_1^{(1)}[n-2k] = & \begin{cases} \phi_{2k+1}[n \bmod N/2] & (2k < N/2 \text{ and } n < N/2) \text{ or} \\ & (2k \geq N/2 \text{ and } n \geq N/2) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6.15}$$

This assignment of the basis functions gives $\mathbf{H}_i^{(1)}$ the required 1-partitionable form. For example, for a set of length $L = 4$ filters, $h_i^{(1)}, i \in \{0, 1\}$, we have

$$\mathbf{H}_i^{(1)} = \begin{pmatrix} \mathcal{H}_i^a + \mathcal{H}_i^b & \mathbf{0} \\ \mathbf{0} & \mathcal{H}_i^a + \mathcal{H}_i^b \end{pmatrix} = \left(\begin{array}{cccc|cccc} h_i[1] & h_i[0] & h_i[3] & h_i[2] & 0 & 0 & 0 & 0 \\ h_i[3] & h_i[2] & h_i[1] & h_i[0] & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & h_i[1] & h_i[0] & h_i[3] & h_i[2] \\ 0 & 0 & 0 & 0 & h_i[3] & h_i[2] & h_i[1] & h_i[0] \end{array}\right). \tag{6.16}$$

It follows that $\mathbf{G}_0^{(1)}\mathbf{H}_0^{(1)} + \mathbf{G}_1^{(1)}\mathbf{H}_1^{(1)} = \mathbf{I}^N$ and $\mathcal{G}_0\mathcal{H}_0 + \mathcal{G}_1\mathcal{H}_1 = \mathbf{I}^{N/2}$, where $\mathbf{G}_i^{(1)} = (\mathbf{H}_i^{(1)})^T$ and $\mathcal{G}_i = \mathcal{H}_i^T$, from the proof in Appendix A..

6.2.4.3 M-partitionable expansions

We extend the 1-PFE to the M-partitionable frequency expansion (M-PFE) as follows:

Definition 3 *A frequency expansion matrix* $\mathbf{H}$ *is* M*-partitionable if the upper left and lower right quadrants are* $(M-1)$*-partitionable.*

Thus, the M-PFE matrices $\mathbf{H}_i^{(M)}$ for $i \in \{0, 1\}$ are defined recursively by

$$\mathbf{H}_i^{(M)} = \begin{pmatrix} \mathbf{H}_i^{(M-1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_i^{(M-1)} \end{pmatrix}. \tag{6.17}$$

We see that $\mathbf{H}_i^{(M)}$ are block diagonal with 2^M partitions. For example, a 3-partitionable matrix has the following form:

$$\mathbf{H}_i^{(3)} = \left(\begin{array}{cc|cc|cc} \mathcal{H}_i & 0 & & & & \\ 0 & \mathcal{H}_i & & 0 & & \\ \hline & & \mathcal{H}_i & 0 & & \\ & 0 & 0 & \mathcal{H}_i & & \mathbf{0} \\ \hline & & & & \mathcal{H}_i & 0 \\ & \mathbf{0} & & & 0 & \mathcal{H}_i \end{array}\right). \tag{6.18}$$

Definition 4 *The frequency expansion (M-PFE) matrix set* $\{\mathbf{H}_0^{(M)}, \mathbf{H}_1^{(M)}\}$ *is* M*-partitionable and orthonormal if* $\{\mathbf{H}_0, \mathbf{H}_1\}$ *satisfy perfect reconstruction condition.*

We note that the Haar FB implements a $(M-1)$-PFE where M is determined from the length of the signal as follows:

Lemma 5 *The Haar filterbank implements a* $(M-1)$*-PFE where* 2^M *is the length of the signal.*

For proof, see Appendix B.

6.2.4.4 M-partitionable convolution

In general, $\mathbf{H}_i$ can be made M-partitionable by wrapping the filters at the $\frac{N}{M+1}$ boundaries (circular convolutions with period $= \frac{N}{M+1}$) as long as the filter length $L \leq \frac{N}{M+1}$. This is equivalent to first segmenting the length N signal into length $\frac{N}{M+1}$ signals and transforming each separately, then concatenating the results. The modified filters $h_i^{(M)}$ and $g_i^{(M)}$ are periodically related to the basis functions by

$$\begin{aligned} h_0^{(M)}[2k-n] = \quad g_0^{(M)}[n-2k] = & \begin{cases} \phi_{2k}[n \bmod \frac{N}{M+1}] & (2k < \frac{N}{M+1} \text{ and } n < \frac{N}{M+1}) \text{ or} \\ & (2k \geq \frac{N}{M+1} \text{ and } n \geq \frac{N}{M+1}) \\ 0 & \text{otherwise} \end{cases} \\ h_1^{(M)}[2k-n] = \quad g_1^{(M)}[n-2k] = & \begin{cases} \phi_{2k+1}[n \bmod \frac{N}{M+1}] & (2k < \frac{N}{M+1} \text{ and } n < \frac{N}{M+1}) \text{ or} \\ & (2k \geq \frac{N}{M+1} \text{ and } n \geq \frac{N}{M+1}) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6.19}$$

This gives the M-PFE, $\mathbf{H}_i^{(M)}$, as follows

$$\mathbf{H}_i^{(M)} = \underbrace{\left(\begin{array}{c|c|c} \begin{array}{c|c} \mathcal{H}_i & 0 \\ \hline 0 & \mathcal{H}_i \end{array} & \ddots & \mathbf{0} \\ \hline \mathbf{0} & \ddots & \mathbf{0} \\ \hline \mathbf{0} & \ddots & \begin{array}{c|c} \mathcal{H}_i & 0 \\ \hline 0 & \mathcal{H}_i \end{array} \end{array}\right)}_{2^M \text{ partitions}}. \tag{6.20}$$

It follows that $\mathbf{G}_0^{(M)}\mathbf{H}_0^{(M)} + \mathbf{G}_1^{(M)}\mathbf{H}_1^{(M)} = \mathbf{I}^N$ and $\mathcal{G}_0\mathcal{H}_0 + \mathcal{G}_1\mathcal{H}_1 = \mathbf{I}^{\frac{N}{M+1}}$, where $\mathcal{G}_i = (\mathcal{H}_i)^T$ and $\mathbf{G}_i^{(M)} = \mathbf{H}_i^{(M)}$ for $i \in \{0,1\}$.

6.2.4.5 M-partitionable expansion (M-PFE) example

We now examine the implications of using the M-PFE and observe that it alters the frequency expansion from the non-partitionable expansion that is derived from the same filter kernel. We examine the following cases in which a four-tap QMF filter constructs the frequency expansions. The filter (QMF4) from [75] is given by

$$h_i = [\ a_i \quad b_i \quad c_i \quad d_i\ ], \tag{6.21}$$

where

$$\begin{array}{llll} a_0 = -0.0915 & b_0 = 0.1585 & c_0 = 0.5915 & d_0 = -0.3415 \\ a_1 = 0.5915 & b_1 = -0.1585 & c_1 = 0.5915 & d_1 = -0.0915. \end{array} \tag{6.22}$$

The "non-partitionable" frequency expansion matrices $\mathbf{H}_i$ are generated from Eq 6.21 and are given by, where $i \in \{0,1\}$ (notice that $\mathbf{H}_i$ has the form depicted in Eq 6.12),

$$\mathbf{H}_i = \left(\begin{array}{cccccccc|cccccccc} a_i & b_i & c_i & d_i & 0 & \cdots & & & & & & & & & & \\ \cdots & 0 & a_i & b_i & c_i & d_i & 0 & \cdots & & & & & & & & \\ & & \cdots & 0 & a_i & b_i & c_i & d_i & 0 & \cdots & & & & & & \\ & & & & \cdots & 0 & a_i & b_i & c_i & d_i & 0 & \cdots & & & & \\ \hline & & & & & & \cdots & 0 & a_i & b_i & c_i & d_i & 0 & \cdots & & \\ & & & & & & & & \cdots & 0 & a_i & b_i & c_i & d_i & 0 & \cdots \\ & & & & & & & & & & \cdots & 0 & a_i & b_i & c_i & d_i \\ c_i & d_i & 0 & \cdots & & & & & & & & & \cdots & 0 & a_i & b_i \end{array}\right). \tag{6.23}$$

Since, the 4-tap QMF filters are linear phase, FIR filters, the h_i are nearly orthogonal and closely approximate the PR condition $\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1 \approx \mathbf{I}$, where $\mathbf{G}_i = \mathbf{H}_i^T$. From Eq 6.21 and Eq 6.15, the 1-PFE matrices $\mathbf{H}_i^{(1)}$ derived from QMF4 are given as

$$\mathbf{H}_i^{(1)} = \left(\begin{array}{cccccccc|cccccccc} a_i & b_i & c_i & d_i & 0 & \cdots & & & & & & & & & & \\ \cdots & 0 & a_i & b_i & c_i & d_i & 0 & \cdots & & & & & & & & \\ & & \cdots & 0 & a_i & b_i & c_i & d_i & & & & & \mathbf{0} & & & \\ c_i & d_i & 0 & & \cdots & 0 & a_i & b_i & & & & & & & & \\ \hline & & & & & & & & a_i & b_i & c_i & d_i & 0 & \cdots & & \\ & & & & \mathbf{0} & & & & \cdots & 0 & a_i & b_i & c_i & d_i & 0 & \cdots \\ & & & & & & & & & & \cdots & 0 & a_i & b_i & c_i & d_i \\ & & & & & & & & c_i & d_i & 0 & & \cdots & 0 & a_i & b_i \end{array}\right). \tag{6.24}$$

We note that by defining a_i, b_i, c_i and d_i as in Eq 6.22, the 1-PFE gives $\mathbf{G}_0^{(1)}\mathbf{H}_0^{(1)} + \mathbf{G}_1^{(1)}\mathbf{H}_1^{(1)} \approx \mathbf{I}$, where $\mathbf{G}_i^{(1)} = (\mathbf{H}_i^{(1)})^T$. Finally, the 2-PFE matrices $\mathbf{H}_i^{(2)}$ are generated from QMF4 using Eq 6.21 and Eq 6.19, which gives

$$\mathbf{H}_i^{(2)} = \left(\begin{array}{c|c|c|c} \begin{matrix} a_i & b_i & c_i & d_i \\ c_i & d_i & a_i & b_i \end{matrix} & \mathbf{0} & \ddots & \\ \hline \mathbf{0} & \begin{matrix} a_i & b_i & c_i & d_i \\ c_i & d_i & a_i & b_i \end{matrix} & \mathbf{0} & \ddots \\ \hline \ddots & \mathbf{0} & \begin{matrix} a_i & b_i & c_i & d_i \\ c_i & d_i & a_i & b_i \end{matrix} & \mathbf{0} \\ \hline & \ddots & \mathbf{0} & \begin{matrix} a_i & b_i & c_i & d_i \\ c_i & d_i & a_i & b_i \end{matrix} \end{array}\right). \tag{6.25}$$

We still approximate PR from the 2-PFE, since, $\mathbf{G}_0^{(2)}\mathbf{H}_0^{(2)} + \mathbf{G}_1^{(2)}\mathbf{H}_1^{(2)} \approx \mathbf{I}$, where $\mathbf{G}_i^{(2)} = (\mathbf{H}_i^{(2)})^T$.

The frequency expansions generated by $\mathbf{H}_i, \mathbf{H}_i^{(1)}$, and $\mathbf{H}_i^{(2)}$, which are all derived from the same nearly-PR filter set (Eq 6.21), generate nearly-PR FBs. We now show that the specific frequency expansions are altered in the PFE forms. Let an impulse signal u be given by $u[0] = 1$ and $\forall_{m\neq 0} u[m] = 0$. Then, $\mathbf{H}_i, \mathbf{H}_i^{(1)}$, and $\mathbf{H}_i^{(2)}$ are defined as above from the filter kernel in Eq 6.22, which gives

$$\begin{array}{lcl} \mathbf{H}_0 u & = & (\ -0.0915 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.5915\ ) \\ \mathbf{H}_0^{(1)} u & = & (\ -0.0915 \quad 0 \quad 0 \quad 0.5915 \quad 0 \quad 0 \quad 0 \quad 0\ ) \\ \mathbf{H}_0^{(2)} u & = & (\ -0.0915 \quad 0.5915 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ) \end{array} \tag{6.26}$$

$$\begin{array}{lcl} \mathbf{H}_1 u & = & (\ -0.3415 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -0.1585\ ) \\ \mathbf{H}_1^{(1)} u & = & (\ -0.3415 \quad 0 \quad 0 \quad -0.1585 \quad 0 \quad 0 \quad 0 \quad 0\ ) \\ \mathbf{H}_1^{(2)} u & = & (\ -0.3415 \quad -0.1585 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ). \end{array} \tag{6.27}$$

We can see that the frequency expansions are modified in $\mathbf{H}_i^{(1)}$ and $\mathbf{H}_i^{(2)}$. Later, we will see that, although the PFE matrices change the frequency expansion, they enable the JASF graph to better adapt to the signal. The overall effect is still to improve compression.

6.2.5 Image expansion

The extension of the frequency expansion to images is accomplished using separable transformations of the rows and columns of the image. This produces a four-channel FB where the output $y_{i,j}$ of channel i, j, where $i, j \in \{0, 1\}$ is given by $y_{i,j} = \mathbf{H}_i x \mathbf{H}_j^T$. The image is resynthesized from the four subbands by

$$\hat{x} = \sum_{i,j} \mathbf{G}_i y_{i,j} \mathbf{G}_j^T = \sum_{i,j} \mathbf{G}_i \mathbf{H}_i x \mathbf{H}_j^T \mathbf{G}_j^T. \tag{6.28}$$

6.3. Image segmentation

The next building block is segmentation. Segmentation is a process by which all data outside of a particular range is set to zero, preserving only a segment of the signal.

6.3.1 Binary segmentation

In general, the binary segmentation of a signal x into two half signals w_0 and w_1 is given by $w_0 = \mathbf{S}_0 x$ and $w_1 = \mathbf{S}_1 x$ where

$$\mathbf{S}_0 = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \mathbf{S}_1 = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I} \end{bmatrix}. \tag{6.29}$$

In the case of segmentation of a signal x of length N, we denote $w_0 = \mathbf{S}_0^N x$ and $w_1 = \mathbf{S}_1^N x$ where

$$\mathbf{S}_0^N = \begin{bmatrix} \mathbf{I}^{N/2} & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \mathbf{S}_1^N = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}^{N/2} \end{bmatrix}, \tag{6.30}$$

where $\mathbf{S}_i^N$ has size $\frac{N}{2} \times \frac{N}{2}$ and $\mathbf{I}^{\mathbf{N/2}}$ is the $\frac{N}{2} \times \frac{N}{2}$ identity matrix. The signal is resynthesized by adding the segments, $\hat{x} = w_0 + w_1 = \mathbf{S}_0^N x + \mathbf{S}_1^N x$. The PR condition is satisfied since the $\mathbf{S}_i's$ satisfy the following

constraint

$$\mathbf{S}_0^N + \mathbf{S}_1^N = \mathbf{I}^N. \tag{6.31}$$

The PR binary segmentation system is illustrated in Figure 6-5.

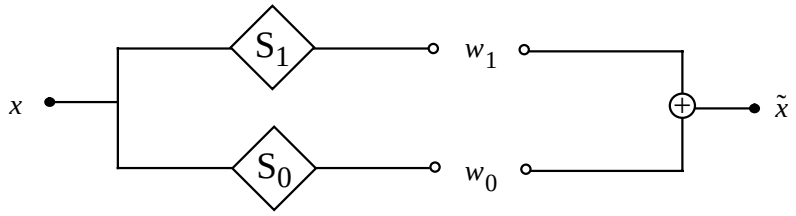


Figure 6-5: Binary segmentation and resynthesis.

6.3.2 Image spatial quadtree (QT) segmentation

The extension of segmentation to images is accomplished using separable segmentation on the rows and columns of the image. This produces a QT spatial segmentation where segment $w_{i,j}$, for $i,j \in \{0,1\}$ is given by $w_{i,j} = \mathbf{S}_i x \mathbf{S}_j^T$. The resynthesis of the image from the four segments is produced by $\hat{x} = \sum_{i,j\in\{0,1\}} y_{i,j} = \sum_{i,j\in\{0,1\}} \mathbf{S}_i x \mathbf{S}_j^T$.

6.4. Combining frequency expansion and segmentation

We now integrate segmentation and PFE to generate the JASF expansion of the image. There are two basic ways to incorporate segmentation into the FB: (1) before analysis frequency expansion and (2) after analysis frequency expansion. In general, these are not equivalent for arbitrary FBs. However, by using PFE's in the FB, an equivalency is guaranteed.

We first examine the case of building a expansion by cascading segmentation and frequency expansion to produce an overall depth = 2 expansion. Then, we illustrate the advantage of using a 1-PFE in the depth = 2 cascade to produce a more compact joint space and frequency expansion.

6.4.1 Non-partitionable expansion cascade

We first note that the frequency expansion matrices $\mathbf{H}_i$ (from Eq. 6.12) can be separated into four quadrants using the segmentation operators $\mathbf{S}_j$ (from Eq. 6.30). For example, the first quadrant of $\mathbf{H}_i$ is segmented by

$$\begin{pmatrix} \mathcal{H}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \mathbf{S}_0^{N/2} \mathbf{H}_i \mathbf{S}_0^N. \tag{6.32}$$

In general, quadrant $j,k \in \{0,1\}$ of $\mathbf{H}_i$ is segmented by

$$\mathbf{H}_i^{j\,k} = \mathbf{S}_j^{N/2} \mathbf{H}_i \mathbf{S}_k^N. \tag{6.33}$$

In the depth = 2 expansion, eight basis elements are generated ($\mathbf{H}_i^{j\,k}$'s), which are defined by Eq 6.33, since, $i,j,k \in \{0,1\}$. From these eight basis elements a variety of expansions and reconstructions of the signal x are generated, such as

- the two half-signals $\mathbf{S}_0 x$ and $\mathbf{S}_1 x$ from $\mathbf{S}_k x = \sum_i \sum_j \mathbf{G}_i \mathbf{H}_i^{j\,k} x$,
- the two signal subbands, $\mathbf{H}_0 x$ and $\mathbf{H}_1 x$ from $\mathbf{H}_i x = \sum_j \sum_k \mathbf{H}_i^{j\,k} x$,
- the segmented subbands, $\mathbf{S}_j^{N/2} \mathbf{H}_i x$, for $i,j \in \{0,1\}$ from $\mathbf{S}_j^{N/2} \mathbf{H}_i x = \sum_k \mathbf{H}_i^{j\,k} x$, and
- the frequency expansions of the half-signals, $\mathbf{H}_i \mathbf{S}_k^N x$, for $i,k \in \{0,1\}$ from $\mathbf{H}_i \mathbf{S}_k^N x = \sum_j \mathbf{H}_i^{j\,k} x$, and
- the reconstructed signal x from $x = \sum_i \sum_j \sum_k \mathbf{G}_i \mathbf{H}_i^{j\,k} x$.

$$\begin{pmatrix} \mathbf{I} \\ \mathbf{H}_0 \\ \mathbf{H}_1 \\ \mathbf{S}_0^{N/2}\mathbf{H}_1 \\ \mathbf{S}_1^{N/2}\mathbf{H}_1 \\ \mathbf{S}_0^{N/2}\mathbf{H}_0 \\ \mathbf{S}_1^{N/2}\mathbf{H}_0 \\ \mathbf{S}_0^N \\ \mathbf{S}_1^N \\ \mathbf{H}_1\mathbf{S}_0^N \\ \mathbf{H}_1\mathbf{S}_1^N \\ \mathbf{H}_0\mathbf{S}_0^N \\ \mathbf{H}_0\mathbf{S}_1^N \end{pmatrix} = \begin{pmatrix} \mathbf{G}_0 & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_1 & \mathbf{G}_0 & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_1 \\ \mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I} \\ \mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{G}_0 & \mathbf{0} & \mathbf{G}_1 & \mathbf{0} & \mathbf{G}_0 & \mathbf{0} & \mathbf{G}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_0 & \mathbf{0} & \mathbf{G}_1 & \mathbf{0} & \mathbf{G}_0 & \mathbf{0} & \mathbf{G}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{S}_0^{N/2}\mathbf{H}_0\mathbf{S}_0^N \\ \mathbf{S}_0^{N/2}\mathbf{H}_0\mathbf{S}_1^N \\ \mathbf{S}_0^{N/2}\mathbf{H}_1\mathbf{S}_0^N \\ \mathbf{S}_0^{N/2}\mathbf{H}_1\mathbf{S}_1^N \\ \mathbf{S}_1^{N/2}\mathbf{H}_0\mathbf{S}_0^N \\ \mathbf{S}_1^{N/2}\mathbf{H}_0\mathbf{S}_1^N \\ \mathbf{S}_1^{N/2}\mathbf{H}_1\mathbf{S}_0^N \\ \mathbf{S}_1^{N/2}\mathbf{H}_1\mathbf{S}_1^N \end{pmatrix} \tag{6.34}$$

These expansions from Eqs 6.33, 6.31 and 6.6 and are summarized in Eq 6.34. The set $\{\mathbf{H}_i^{jk}\}$, which are generated from the non-partitionable frequency expansion are of the type of basis elements that are generated in the time-frequency tree explored in [63].

6.4.2 Partitionable expansion cascade

We now examine the advantage of using 1-PFEs in the depth = 2 expansion. It follows from Eqs 6.33 and 6.13 that if $\mathbf{H}_i^{(1)}$ are 1-PFE because we have $\mathbf{S}_j^{N/2}\mathbf{H}_i^{(1)}\mathbf{S}_k^N = \mathbf{0}$ for $j \neq k$. This given by Definition 1, which requires that the upper-right ($\mathbf{S}_0^{N/2}\mathbf{H}_i^{(1)}\mathbf{S}_1^N = 0$) and lower-left ($\mathbf{S}_1^{N/2}\mathbf{H}_i^{(1)}\mathbf{S}_0^N = 0$) quadrants of $\mathbf{H}_i^{(1)}$ are zero. This reduces the number of non-zero basis elements ($\mathbf{H}_i^{jk}$'s) from eight to four. As a result, Eq. 6.34 takes the modified form illustrated in Eq. 6.35. In other words, now from only the four basis elements ($\mathbf{S}_j^{N/2}\mathbf{H}_i^{(1)}\mathbf{S}_j^N$, for $i, j \in \{0, 1\}$), the same previous variety of expansions of the signal that were generated from the eight "non-partitionable" basis elements are formed. Notice that commutativity of segmentation and frequency

$$\begin{pmatrix} \mathbf{I} \\ \mathbf{H}_0^{(1)} \\ \mathbf{H}_1^{(1)} \\ \mathbf{S}_0^{N/2}\mathbf{H}_1^{(1)} \\ \mathbf{S}_1^{N/2}\mathbf{H}_1^{(1)} \\ \mathbf{S}_0^{N/2}\mathbf{H}_0^{(1)} \\ \mathbf{S}_1^{N/2}\mathbf{H}_0^{(1)} \\ \mathbf{S}_0^N \\ \mathbf{S}_1^N \\ \mathbf{H}_1^{(1)}\mathbf{S}_0^N \\ \mathbf{H}_1^{(1)}\mathbf{S}_1^N \\ \mathbf{H}_0^{(1)}\mathbf{S}_0^N \\ \mathbf{H}_0^{(1)}\mathbf{S}_1^N \end{pmatrix} = \begin{pmatrix} \mathbf{G}_0^{(1)} & \mathbf{G}_1^{(1)} & \mathbf{G}_0^{(1)} & \mathbf{G}_1^{(1)} \\ \mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{G}_0^{(1)} & \mathbf{G}_1^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_0^{(1)} & \mathbf{G}_1^{(1)} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{S}_0^{N/2}\mathbf{H}_0^{(1)}\mathbf{S}_0^N \\ \mathbf{S}_0^{N/2}\mathbf{H}_1^{(1)}\mathbf{S}_0^N \\ \mathbf{S}_1^{N/2}\mathbf{H}_0^{(1)}\mathbf{S}_1^N \\ \mathbf{S}_1^{N/2}\mathbf{H}_1^{(1)}\mathbf{S}_1^N \end{pmatrix} \tag{6.35}$$

expansion is now provided using 1-PFE, because for $i, j \in \{0, 1\}$ we have,

$$\mathbf{S}_j^{N/2}\mathbf{H}_i^{(1)} = \mathbf{H}_i^{(1)}\mathbf{S}_j^N = \mathbf{S}_j^{N/2}\mathbf{H}_i^{(1)}\mathbf{S}_j^N. \tag{6.36}$$

The commutativity of segmentation and 1-PFE allows the graph structure to generate the complete joint expansion rather than requiring separate tree structures, as in [63], as we see shortly.

6.4.3 Two-step expansion cascades

We first look in more detail at the process of integrating the segmentation and frequency expansion in the depth = 2 cascade. There are four cases that correspond to the four unique, four-step paths that decompose signal x into four basis elements and reconstruct x as illustrated in Figure 6-6. Since the PR condition of the filter bank and segmentation are satisfied, the following two four-step cascades: $F^{(1)} \rightarrow S \rightarrow S^{-1} \rightarrow (F^{(1)})^{-1}$ and $S \rightarrow F^{(1)} \rightarrow (F^{(1)})^{-1} \rightarrow S^{-1}$ reconstruct the signal. We now address the other two cases and demonstrate that reconstruction is provided regardless of the orders of the segmentation and 1-PFE operations in the analysis and synthesis paths.

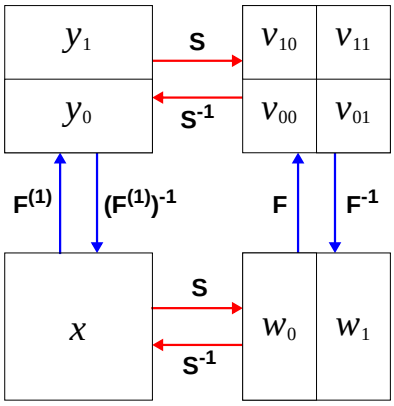


Figure 6-6: Hybrid cascades with frequency expansion and segmentation ($F^{(1)} = \{\mathbf{H}_0^{(1)}, \mathbf{H}_1^{(1)}\}$, and $(F^{(1)})^{-1} = \{\mathbf{G}_0^{(1)}, \mathbf{G}_1^{(1)}\}$).

6.4.3.1 Case 1: $F^{(1)} \rightarrow S \rightarrow (F^{(1)})^{-1} \rightarrow S^{-1}$

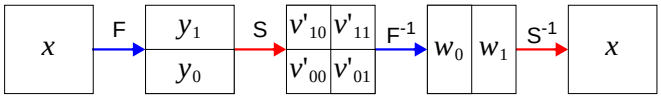


Figure 6-7: Hybrid cascade case 1: $F^{(1)} \rightarrow S \rightarrow (F^{(1)})^{-1} \rightarrow S^{-1}$.

The FB and segmentation are cascaded such that the orders within the analysis and synthesis cascades are not mirrors, as depicted in Figure 6-7 and Figure 6-8. First, we see that the two-step cascade $F^{(1)} \rightarrow S$ produces four pieces, v'_{ij}, where $i, j \in \{0, 1\}$ given by

$$v'_{ij} = \mathbf{S}_j^{N/2} y_i = \mathbf{S}_j^{N/2} \mathbf{H}_i^{(1)} x. \tag{6.37}$$

The resynthesis of the original signal by $(F^{(1)})^{-1} \rightarrow S^{-1}$ is possible because $\mathbf{H}_i^{(1)}$ are 1-PFE. From the commutativity of segmentation and 1-PFE in Eq 6.36, it follows that $v'_{ij} = \mathbf{H}_i^{(1)} \mathbf{S}_j^N$. In this case, the signal is reconstructed by (1) 1-PFE synthesis and (2) adding the appropriate basis elements, as follows,

$$\begin{aligned} \hat{x} &= \textstyle\sum_i \sum_j \mathbf{G}_i^{(1)} v'_{ij} \\ &= \mathbf{G}_0^{(1)} v'_{00} + \mathbf{G}_1^{(1)} v'_{10} + \mathbf{G}_0^{(1)} v'_{01} + \mathbf{G}_1^{(1)} v'_{11} \\ &= w_0 + w_1 = (\mathbf{S}_0^N + \mathbf{S}_1^N) x = x. \end{aligned} \tag{6.38}$$

6.4.3.2 Case 2: $S \rightarrow F^{(1)} \rightarrow S^{-1} \rightarrow (F^{(1)})^{-1}$

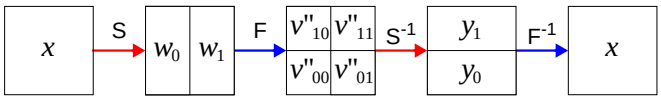


Figure 6-8: Hybrid cascade case 2: $S \rightarrow F^{(1)} \rightarrow S^{-1} \rightarrow (F^{(1)})^{-1}$.

By segmenting before the 1-PFE as illustrated in Figure 6-8, the two-step cascade $S \rightarrow F^{(1)}$ produces four pieces, v''_{ij}, where $i, j \in \{0, 1\}$ given by

$$v''_{ij} = \mathbf{H}_i^{(1)} \mathbf{S}_j^N x. \tag{6.39}$$

The resynthesis of the original signal by $S^{-1} \rightarrow (F^{(1)})^{-1}$ is possible only because $\mathbf{H}_i^{(1)}$ are are 1-partitionable. From the commutativity of segmentation and 1-PFE in Eq 6.36 it follows that $v''_{ij} = \mathbf{S}_j^{N/2} \mathbf{H}_i^{(1)}$. In this case,

the signal is reconstructed by (1) adding the appropriate basis elements and (2) 1-PFE synthesis, as follows,

$$\begin{aligned} \hat{x} &= \sum_i \mathbf{G}_i^{(1)} \sum_j v''_{ij} \\ &= \mathbf{G}_0^{(1)}(v''_{00} + v''_{01}) + \mathbf{G}_1^{(1)}(v''_{10} + v''_{11}) \\ &= \mathbf{G}_0^{(1)} y_0 + \mathbf{G}_1^{(1)} y_1 = (\mathbf{G}_0^{(1)}\mathbf{H}_0^{(1)} + \mathbf{G}_1^{(1)}\mathbf{H}_1)x = x. \end{aligned} \tag{6.40}$$

6.4.4 Multiple step expansion cascades

We generalize to an arbitrary-depth joint expansion. By using k-PFEs, the basis elements are generated by following any of the number of paths that lead to that node in the JASF graph. For example, in Figure 6-9, this requires that the frequency expansion F^k is carried out with a FB that implements a k-PFE $\mathbf{H}_0^{(k)}$ and $\mathbf{H}_1^{(k)}$.

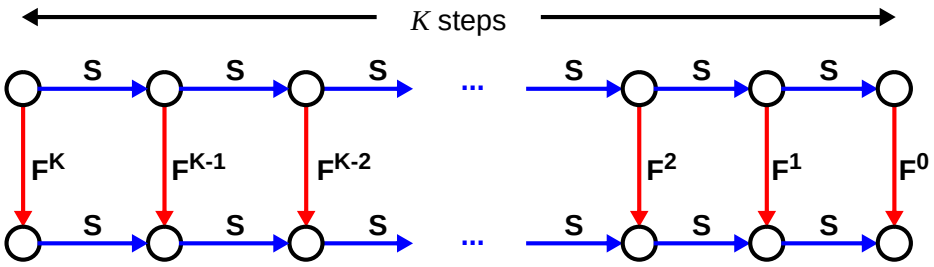


Figure 6-9: K step expansion cascade in space and one step in frequency.

Figure 6-10 illustrates the depth = M, JASF graph which combines segmentations (S) and k-PFEs (F^k).

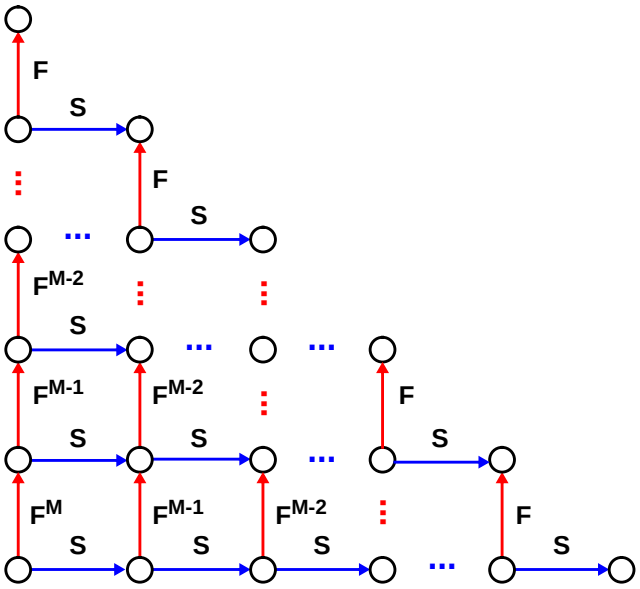


Figure 6-10: JASF graph combines k-PFE filter banks ($F^k = \{\mathbf{H}_0^{(k)}$ and $\mathbf{H}_1^{(k)}\}$) and segmentation (S).

6.5. Tree and graph structured expansions

We adopt the following notation to illustrate the tree- and graph-structured expansions: Figure 6-11(a) summarizes the WP-tree depicted in Figure 6-12, and Figure 6-11(b) summarizes the QT depicted in Figure 6-13. In other words, in Figure 6-11 each straight-line corresponds to an embedded tree. We look in more detail at each tree and graph.

6.5.1 Complexity

We evaluate the tree- and graph-structured expansions in the (1) sizes of the basis element libraries (N), (2) the number of bases (B), and (3) the number of over-expansions of the image that are required to generate the basis element libraries (E). The depth of the trees and graphs are given by D, and $\beta =$ gives the splitting factor. The splitting factor determines the number of children nodes produced in a single frequency expansion or segmentation. For example, both frequency expansion and QT image segmentation produce $\beta = 4$ children.

(a) (b)

Figure 6-11: Using compact notation, (a) WP-tree, and (b) spatial QT, where the numbers $(1, 4, 16, \ldots)$ give the number of nodes (or basis elements) at each level. The number of nodes will be omitted in later figures.

The number of basis elements in the library (N), number of unique bases (B), and number of over expansions (E) that are generated by the tree- and graph-structured expansions are summarized in Table 6.5.1.

Basis element library sizes for 2-D image with $\beta = 4$ (splitting factor)

	Single-Tree			DT			Redundant Tree			JASF Graph		
D	N_s	E_s	B_s	N_d	E_d	B_d	N_r	E_r	B_r	N_g	E_g	B_g
1	1	1	1	1	1	1	1	1	1	1	1	1
2	5	2	2	9	3	3	9	5	3	9	3	3
3	21	3	17	57	6	98	73	21	163	57	6	162
4	85	4	83522	313	10	10^7	585	85	10^9	313	10	10^9
5	341	5	10^{19}	1593	15	10^{31}	4681	341	10^{36}	1593	15	10^{36}
6	1365	6	10^{78}	7737	21	10^{127}	37449	1365	10^{147}	7737	21	10^{147}

6.5.2 Single-trees

Single-tree expansions are generated by the WP-tree and QT. Where $N \times N$ is the size of the image, the single-trees can be grown to arbitrary depth, $D \leq N \log_2 N$. The single-tree complexity is given by the following recursive relations

$$\begin{array}{lll} N_s(D) = & 1 + \beta N_s(D-1), & N_s(0) = 0 \\ E_s(D) = & 1 + E_s(D-1), & E_s(0) = 0 \\ B_s(D) = & 1 + B_s(D-1)^\beta, & B_s(0) = 0. \end{array} \tag{6.41}$$

These are derived as follows: N_s – each depth $= D$ tree consists of one node plus β sub-trees of depth $= D-1$. An empty single-tree, $D = 0$, has no nodes. E_s – each level of the tree produces one expansion of the signal. B_s – each node in the tree can be represented by itself or by its β children. This recursive definition gives B_s.

A single-tree expansion of a 512×512 image, of depth $D = 5$, with $\beta = 4$, generates $N_s = 341$ basis elements, expands the image $E_s = 5$ times and generates a library of $B_s \approx 10^{19}$ bases. We now examine the two types of single-trees: WP-tree and QT.

6.5.2.1 Wavelet packet (WP)-tree

Arbitrary FBs can be produced by cascading the two-channel FBs. Since the two-channel FB implements an orthonormal frequency expansion, any cascade is orthonormal since it merely iterates the expansion. In other words, the impulse responses of the overall filters and their appropriate shifts also form an orthonormal basis for $l_2(z)$ [167]. The transfer function of an arbitrary cascade of filters and downsamplers is given by,

$$H_r(z) = \prod_{k=0}^{d_r - 1} H_{P_r(k)}(z^{2^k}), \tag{6.42}$$

which is followed by downsampling by 2^{d_r}. Here $d_r =$ depth of the cascade of path r, and $P_r(k) \in \{0, 1\}$ is an indicator function that selects the filter, $\mathbf{H}_0$ or $\mathbf{H}_1$, at stage k in the filter path. For example, the two-channel

FBs can be cascaded into a tree-structure as depicted in Figure 6-12.

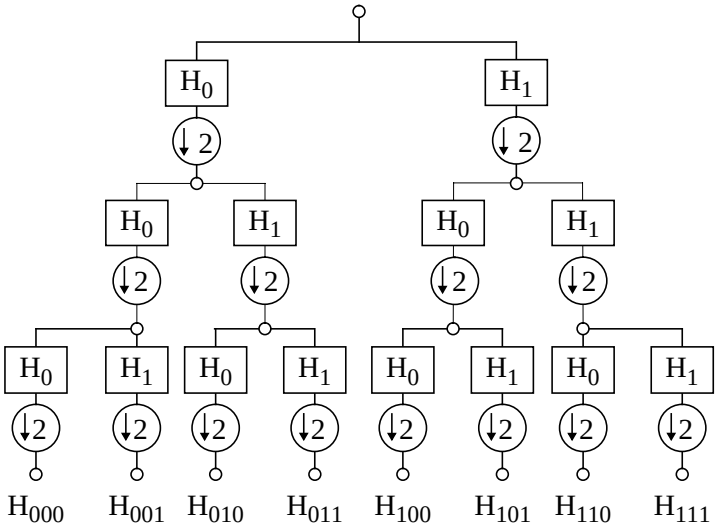


Figure 6-12: Tree-structured filter bank (FB).

6.5.2.2 Spatial quad-tree (QT)

The signal is segmented by cascading the binary segmentations in a tree-structure. For example, a full tree cascade is depicted in Figure 6-13, whereby the leaf nodes of the tree give the segments of the signal. The overall segmentation, $\mathbf{S}_r^N$, of a length N signal produced by a path r of arbitrary segmentations is given by $\mathbf{S}_r^N = \mathbf{S}_{P_r}^N$ where

$$\mathbf{S}_{P_r,0}^N = \begin{bmatrix} \mathbf{S}_{P_r}^{N/2} & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \mathbf{S}_{P_r,1}^N = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{S}_{P_r}^{N/2} \end{bmatrix}, \tag{6.43}$$

where $P_r(k) \in \{0, 1\}$ is an indicator function that selects segmentation $\mathbf{S}_0$ or $\mathbf{S}_1$ at stage k of the segmentation path. PR of the signal from a complete set of arbitrary segments is provided by the recursive relations $\mathbf{S}_{P_r}^N = \mathbf{S}_{P_r,0}^{N/2} + \mathbf{S}_{P_r,1}^{N/2}$ since

$$\mathbf{S}_{P_r}^N = \begin{bmatrix} \mathbf{S}_{P_r,0}^{N/2} & 0 \\ 0 & \mathbf{S}_{P_r,1}^{N/2} \end{bmatrix}. \tag{6.44}$$

6.5.3 Double-tree (DT)

The insufficiency of the single-tree adaptive expansions – WP-tree and QT is addressed in [62]. The authors proposed the DT which combines binary segmentation and WP-trees. As illustrated in Figure 6-14(a), the DT generates a hierarchy of dyadic segmentations and WP expansions of each segment. Figure 6-14(a) also illustrates that the DT treats the frequency expansion and segmentation asymmetrically.

For example, the DT does not segment any of the frequency F nodes in the tree. In other words, segmentation never follows frequency expansion. This limits the number of bases. We see that a single critical pruning at the root of the DT in which segmentation is not selected reduces the DT to the WP-tree.

6.5.4 Dual double tree (DDT)

The dual of the DT also combines QT and WP-trees. The dual double tree (DDT) is produced by first growing a single WP-tree followed by the segmentation of the WP-tree basis elements. The DDT is depicted in Figure 6-14(b). The complexities of the DT and DDT are given by the following recursive relations

$$\begin{array}{lll} N_d(D) = & N_s(D) + \beta N_d(D-1), & N_d(0) = 0 \\ E_d(D) = & E_s(D) + E_d(D-1), & E_d(0) = 0 \\ B_d(D) = & 1 + B_s(D-1)^\beta + B_d(D-1)^\beta, & B_d(0) = 0. \end{array} \tag{6.45}$$

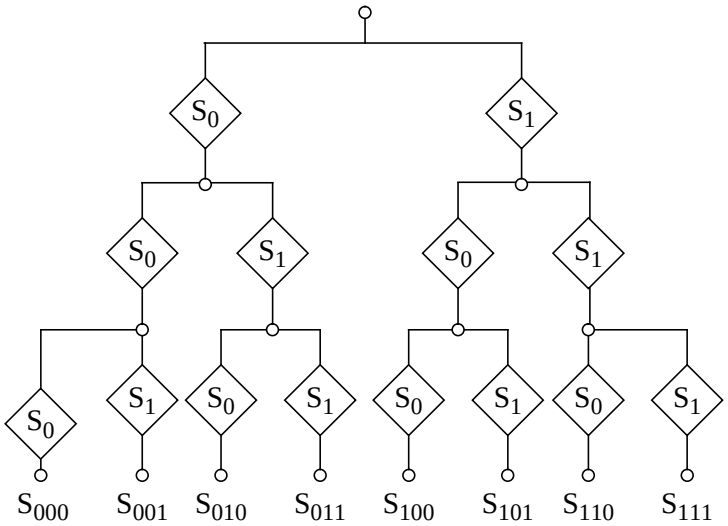


Figure 6-13: Tree-structured segmentation.

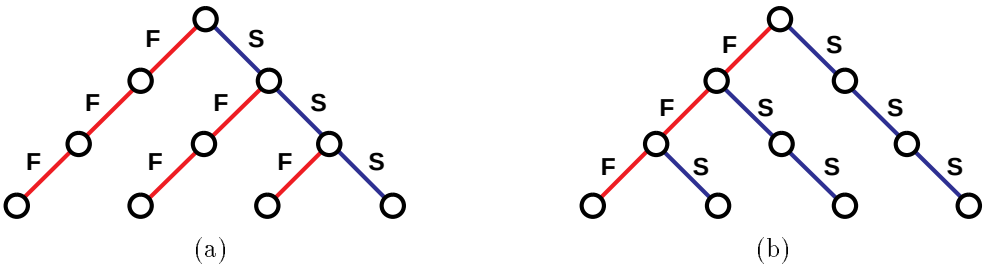


Figure 6-14: Asymmetric joint space and frequency expansions (a) DT, (b) DDT

These are derived as follows: N_d – each depth $= D$, DT consists of one depth D single-tree plus β DTs of depth $D-1$. An empty DT, $D=0$, has no basis elements. E_d – at each level $= D$ in the DT, there is one depth $= D$, single-tree and one depth $= D-1$ DT. B_d – at each node at level $= D$ in the DT, there are three options for basis selection, (1) prune S and F, which gives only one basis of the node, (2) prune S, which gives β single-trees of depth $= D-1$ that provides $B_s(D-1)^\beta$ bases, and (3) prune F, which gives β DTs of depth $= D-1$ that provides $B_d(D-1)^\beta$ bases.

A DT or DDT expansion of a 512×512 image, of depth $D=5$, with $\beta = 4$, generates $N_d = 1593$ basis elements, expands the image $E_d = 15$ times and generates a library of $B_s \approx 10^{31}$ bases.

6.5.5 Redundant space and frequency tree (RSFT)

As illustrated in Figure 6-14, both the DT and the DDT provide an asymmetric treatment of segmentation and frequency expansion. In order to produce the symmetric expansion, all nodes in the trees need both frequency expansion and segmentation. In straightforward implementation, this generates the much larger tree structure, which is illustrated in Figure 6-15. This redundant space and frequency tree (RSFT) combines both the DT and the DDT.

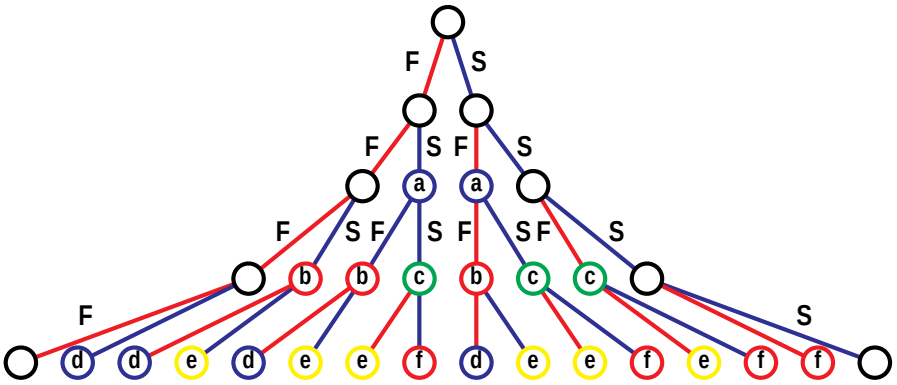


Figure 6-15: Redundant space and frequency tree (RSFT) provides a symmetric expansion in segmentation and frequency operations.

The complexity of the RSFT is given by the following recursive relations

$$\begin{array}{lll} N_r(D) = & 1 + 2\beta N_r(D-1), & N_r(0) = 0 \\ E_r(D) = & 1 + 2E_r(D-1), & E_r(0) = 0 \\ B_r(D) = & 1 + 2B_r(D-1)^\beta, & B_r(0) = 0, B_r(1) = 1. \end{array} \tag{6.46}$$

These are derived as follows: N_r – each depth $= D$, RSFT consists of one basis element plus β RSFTs of depth $= D-1$ from the frequency-expansion, plus β RSFTs of depth $= D-1$ from the segmentation. An empty RSFT, $D=0$, has no basis elements. E_r – each node is expanded by segmentation and frequency expansion. This gives twice the number of expansions at level $= D$ than level $= D-1$. B_r – at each node at level $= D$ in the DT, there are three options for basis selection, (1) prune S and F, which gives only one basis of the node, (2) prune S, which gives β RSFTs of depth $= D-1$ that provides $B_r(D-1)^\beta$ bases, and (3) prune F, which gives another β RSFTs of depth $= D-1$ that provides $B_r(D-1)^\beta$ bases.

A RSFT expansion of a 512×512 image, of depth $D=5$, with $\beta = 4$, generates $N_r = 4681$ basis elements, expands the image $E_r = 341$ times and generates a library of $B_r \approx 10^{36}$ bases.

6.5.6 Joint adaptive space and frequency (JASF) graph

The redundancy in the RSFT (see Figure 6-16) is removed by using partitionable expansions and the JASF graph [140]. When the frequency expansion is inherently partitionable (i.e., Haar filter bank has $\mathbf{H}_i$'s which are already M-PFE), the JASF graph basis library includes all bases attainable by the RSFT. In this case, the RSFT is clearly redundant in that it generates a larger library than the JASF graph, that is $N_r(D) \geq N_g(D)$. When using non-partitionable frequency expansions in the RSFT, there still exists "near-redundancy" in the RSFT library. We will show that these additional RSFT library elements provide little gain in compression performance.

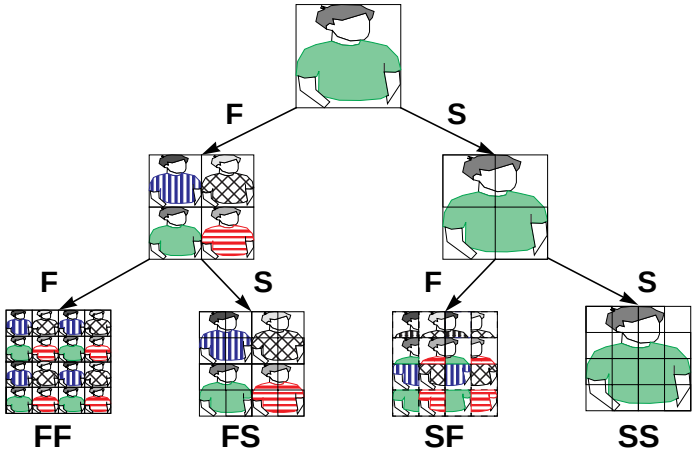


Figure 6-16: Example RSFT expansion of an image. In the RSFT, the basis elements generated by the $F \rightarrow S$ branch are approximately equivalent to those generated by the $S \rightarrow F$ branch.

Since the output of the $S \rightarrow F^{(1)}$ cascade is equivalent to the output of the $F^{(1)} \rightarrow S$ cascade, the data does not need to be produced twice. As illustrated for the image in Figure 6-16, the expansions generated from the $S \rightarrow F^{(1)}$ cascade is the same as that in the $F^{(1)} \rightarrow S$ cascade, but the order is different. When the RSFT is modified to collect the redundant nodes, the JASF graph structure is generated, see Figure 6-17. Inspection reveals that by using M-PFEs, the DT, Figure 6-14(a), the DDT, Figure 6-14(b), and the JASF graph, Figure 6-17, have identical basis elements. The JASF graph offers the maximum selection from the basis elements to generate bases.

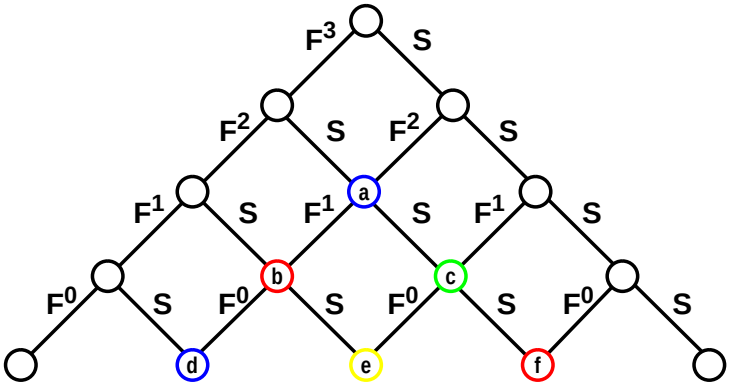


Figure 6-17: The JASF graph provides symmetric expansion in space and frequency, and avoids the "near" redundancy of the RSFT.

The complexity of the JASF graph is given by the following recursive relations

$$\begin{array}{lll} N_g(D) = & N_s(D) + \beta N_g(D-1), & N_g(0) = 0 \\ E_g(D) = & E_s(D) + E_g(D-1), & E_g(0) = 0 \\ B_g(D) = & 1 + 2B_g(D-1)^{\beta} - B_g(D-2)^{\beta^2}, & B_g(0) = 0, B_g(1) = 1. \end{array} \tag{6.47}$$

These are derived as follows: N_g – each depth $= D$, JASF graph consists of one depth $= D$ single-tree, plus β JASF graphs of depth $= D-1$. An empty JASF graph, $D = 0$, has no basis elements. E_g – at each level $= D$ in the JASF graph, there is one depth $= D$ single-tree and one DT of depth $= D-1$. B_g – at each node at level $= D$ in the DT, there are three options for basis selection, (1) prune S and F, which gives only one basis of the node, (2) prune S, which gives β JASF graphs of depth $= D-1$ that provides $B_g(D-1)^{\beta}$ bases, and (3) prune F, which gives β JASF graphs of depth $= D-1$ that provides $B_g(D-1)^{\beta}$ bases. The options (2) and (3) have one JASF graph of depth $= D-2$ in common, which requires the reduction by $B_g(D-2)^{\beta^2}$

bases in the computation.

A JASF graph expansion of a 512×512 image, of depth $D = 5$, with $\beta = 4$, generates $N_g = 1593$ basis elements, expands the image $E_g = 15$ times and generates a library of $B_g \approx 10^{36}$ bases.

6.6. Basis element library

In order to select the elements from the JASF library. the nodes in the JASF graph are analyzed and compared. Each JASF graph node is indexed by its graph position which defines a series of frequency and segmentation operations that generated that node. Equivalently, each JASF library element is indexed by its location and size in the space-frequency space. For example, in the 1-D version of the JASF graph, each element (region in the time-frequency plane) is indexed by (i, j, k, l), where i = time resolution, j = frequency resolution, k = position in time and l = position in frequency. For images, the JASF graph generates an eight-index transform: (i_x, j_x, k_x, l_x) and (i_y, j_y, k_y, l_y), which includes x and y directions in space and frequency.

6.6.1 Basis sets

To generate a basis, the basis elements in the library are selected such that the image can be reconstructed completely and non-redundantly. Using the indexing notation of the time-frequency plane, this requires that elements $t_{i,j,k,l}$'s generates the basis $\{T^*_{i,j,k,l}\}$ by forming a tiling of the time-frequency plane as follows

$$\bigcup_{t_{i,j,k,l} \in \{T^*_{i,j,k,l}\}} t_{i,j,k,l} \supseteq l^2(z). \tag{6.48}$$

We can show that this tiling or completeness requirement is satisfied only if

$$\sum_{t_{i,j,k,l} \in \{T^*_{i,j,k,l}\}} 2^{-i-j} = 1. \tag{6.49}$$

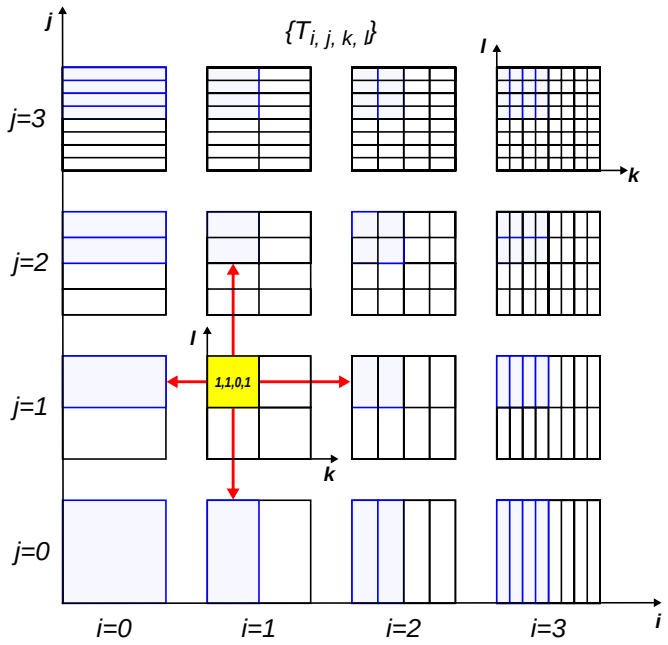


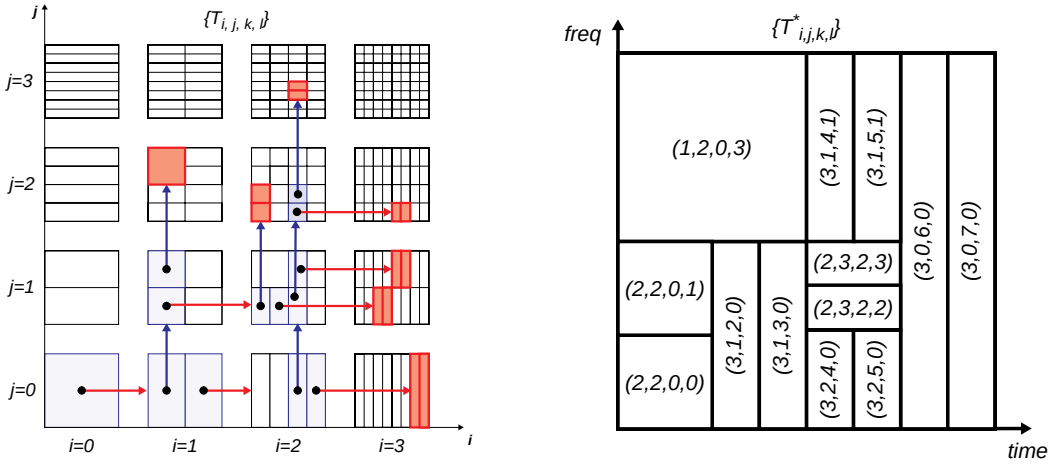
Figure 6-18: Selection of node $t_{1,1,0,1}$ excludes all darkened nodes from the basis.

If the basis elements do not overlap, Eq 6.49 guarantees that the basis is complete (though possibly over-complete). To ensure that the basis is non-redundant – no overlap, each basis element included in the basis necessarily excludes other elements. For example, this is illustrated in Figure 6-18. When basis element $t_{1,1,0,1}$ is included in the basis, in order to not overlap with other basis elements, all the darkened elements

are excluded. This requires that basis element $t_{i,j,k,l}$ is in the basis $\{T^*\}$ if and only if $t_{i',j',k',l'}$ is not in the basis $\{T^*\}$ where $i', j' \in z, k2^{i'-i} \leq K < k2^{i'-i+1}$ and $l2^{j'-j} \leq L < l2^{j'-j+1}$.

The restrictions on basis membership are isomorphic to conditions on the JASF graph. In the JASF graph, a basis is given by a set of terminal nodes in a tree that is embedded in the JASF graph. For example, Figure 6-19(a) indicates a selection of basis elements from the library (or nodes from the JASF graph) that generate the tiling (basis) in Figure 6-19(b). By pruning the JASF graph, the selection of basis elements satisfies the constraints of completeness and non-redundancy.

An example of JASF graph basis selection is depicted in Figure 6-19(a). In this example, the light shaded nodes are the intermediate nodes in the pruned graph, the dark nodes are the terminal nodes and the unshaded nodes have been pruned from the graph. The basis is given by the set of terminal nodes. The basis generated in this example (see Figure 6-19) is not accessible from the WP-tree, the spatial QT, the DT or the DDT.



(a) (b)

Figure 6-19: Example of basis selection which cannot be obtained via the DT or WP-tree. (a) A basis from dyadic time/frequency library, (b) corresponding time/frequency tiling.

6.6.2 Basis selection

The selection of the basis from the JASF library involves a three-way decision at each node: (1) prune F and S, (2) prune S, or (3) prune F. This is a simple extension of the fast basis selection algorithm developed in [31] and [124], which involves a two-way decision at each node: (1) prune, (2) not-prune. The JASF basis search works as follows:

1. a coding cost (J_i) is assigned to each basis element in the JASF graph basis library (= each node in the JASF graph).
2. Starting from the root node, recursively, the least cost embedded tree from each node is found by three-way pruning, select $\min(J_i, \sum J_{i,f_k}, \sum J_{i,s_k})$.
3. The embedded tree after all pruning gives the basis with the lowest total cost.

6.6.3 Rate-distortion optimization

In order to produce the basis with the lowest total coding cost, a cost function indicates the cost of coding each node. One restriction is that the cost function is additive [124]. Coifman and Meyer [32] use the entropy ($J_i = R_i$) of each node as the cost function. Ramchandran and Vetterli [124] demonstrate that a rate-distortion cost function is more suited to the compression problem. The two-sided measure ($J_i = R_i + \lambda D_i$) better matches the goal of finding the optimal rate-distortion point for compression [124]. We assign the rate-distortion costs (J_i) to each node in the graph by quantizing using a set of quantizers. This produces a set of rate-distortion points (R_i, D_i) for each node. The procedure for pruning the graph involves the simultaneous identification of the optimal rate-distortion points (R_i^*, D_i^*) and the pruning decisions. For a detailed description of the algorithm, refer to [124].

A brief description follows: the algorithm sweeps through a series of rate-distortion trade-off values; λ. For each λ. the least cost quantizer for each node is selected as the one that minimizes $J_i^* = D_i^* + \lambda R_i^*$. Next, the least cost embedded tree is found by pruning. The total rate $\sum R_i^*$ is compared to the budget R_T. If the total rate exceeds the budget, $\sum R_i \geq R_T$, then λ is adjusted and the procedure is repeated. The subsequent values of λ are selected using Newton's method. The iteration process converges to the best basis (lowest distortion) within the constrain of the bit budget R_T.

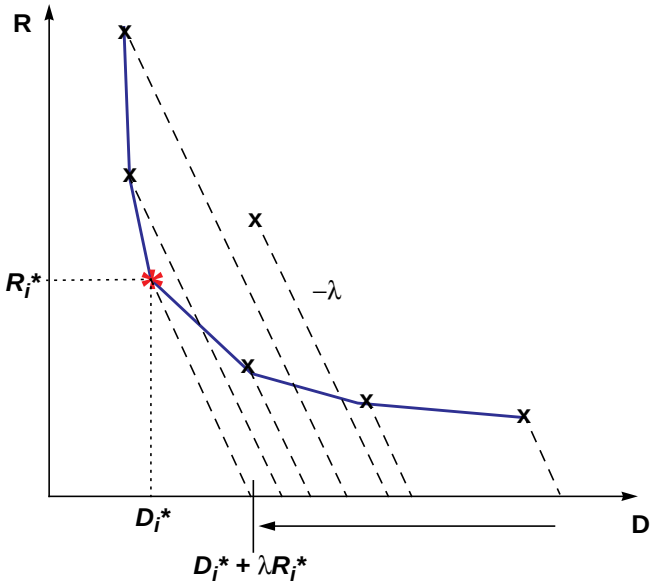


Figure 6-20: Rate-distortion operating points (R_i, D_i) and optimal R-D point selection via trade-off constraint λ.

6.7. Image compression examples

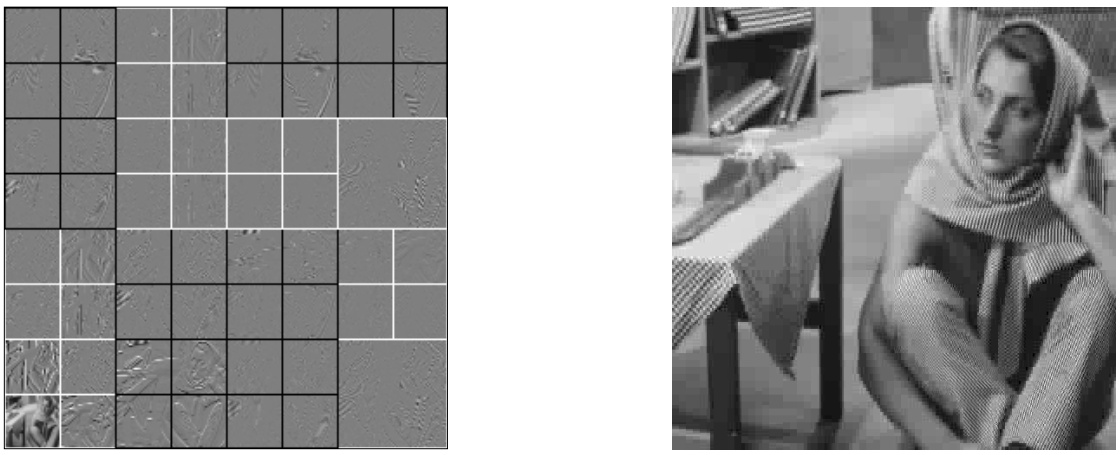
We evaluate the spatial QT, WP-tree, DT, JASF graph and RSFT image expansions by examining their image compression performance. For each expansion-type, the best basis is selected from the library that minimizes the rate-distortion coding cost, $D + \lambda R$, within the constraints of a total bit-rate, R_T.

6.7.1 Haar filter JASF graph

In this example, the two-tap Haar filter $h_0 = [1, 1]$ is used. In this case, all frequency expansions for the WP-tree, DT, DDT and JASF graph are carried out in a partitionable-form (because the Haar FB is partitionable, see Appendix B.). The results, depicted in Table 6.1, show improved compression performance by the JASF graph over spatial QT, WP-tree, and DT methods, and other compression methods such as JPEG and the Haar wavelet transform. The optimal bases chosen by the JASF graph are not accessible by the spatial QT, WP-tree, or DT. The best JASF graph basis for the target bit-rate of 0.25 bpp is depicted in Figure 6-21(a). The encoded images is shown in Figure 6-21(b).

	JPEG	wavelet transform	spatial QT	WP-tree	DT	JASF graph
0.5 bpp	28.3 db	29.5 db	19.1 db	32.7 db	32.7 db	**33.0** db
1.0 bpp	33.1 db	34.6 db	26.5 db	37.1 db	37.1 db	**37.7** db
2.0 bpp	38.9 db	40.7 db	34.7 db	43.0 db	43.0 db	**43.8** db

Table 6.1: Compression results on the *Barbara* image using Haar filter.



(a) (b)

Figure 6-21: JASF compression, (a) JASF graph basis for *Barbara* image, (b) reconstructed at 0.25 bpp.

6.7.2 QMF JASF graph

In this example, the twelve-tap QMF filter (QMF12a) from [75] is used. For the JASF graph, the partitionable-form frequency expansion matrices are constructed from the QMF12a filter kernel. The results, depicted in Table 6.2, show improved compression performance by the JASF graph over the spatial QT, WP-tree and DT methods. Again, the optimal bases chosen by the JASF graph are not accessible by the spatial QT, WP-tree or DT. We see also that the JASF graph approximates the performance of the RSFT. The additional elements in the RSFT library do not provide for significant improvement in compression performance. The partitionable-form of the frequency expansion does not decrease the image compression performance.

	spatial QT	WP tree	DT	JASF graph	RSFT
0.25 bpp	N/A	27.9 db	27.9 db	**28.4** db	**28.4** db
0.5 bpp	19.0 db	32.3 db	32.3 db	**32.7** db	**32.7** db
1.0 bpp	25.1 db	36.8 db	36.8 db	**37.5** db	**37.5** db
2.0 bpp	33.1 db	42.9 db	42.9 db	**43.8** db	**43.8** db

Table 6.2: Compression results on the *Barbara* image using QMF12a filter.

6.8. Summary

We presented a new type of adaptive image expansion (JASF graph) and a joint adaptive space and frequency image basis selection method. The JASF generates an efficient symmetric expansion by combining "partitionable" frequency expansion with spatial segmentation. The JASF library provides a greater number of bases than recent adaptive- wavelet packet, spatial quad-tree and the double-tree methods. Image compression evaluations demonstrate improved adaptability to the images and improved compression performance using the JASF graph.

Chapter 7

Spatial and Feature Query

7.1. Introduction

In this chapter, we present the strategies for computing queries that specify the features, sizes and arbitrary spatial layouts of regions, which include both absolute and relative spatial locations. The spatial and feature system integrates content-based querying with spatial query techniques to provide a new paradigm for searching for images by arrangements of regions.

We also address several special case spatial queries involving adjacency, overlap and encapsulation of regions. Finally, we evaluate the feature and spatial queries and demonstrate the improved image query capabilities over non-spatial content-based approaches.

The spatial and feature query system is composed of four sub-systems: (1) the image analysis subsystem, (2) the query engine, (3) the user-interface and (4) the image database. These sub-systems are combined to provide a complete solution for the search and retrieval of images. We present the design of the separate components of the system and evaluate the overall performance.

We demonstrate examples of spatial and feature queries on several databases of images, including symbolic images, synthetic image and photographic color images [146]. In particular, we demonstrate that the spatial and feature query paradigm provides a powerful technique for image retrieval and improves performance over traditional content-based image query systems.

7.1.1 Integrating spatial and feature query

The integrated spatial and feature query method allows users to flexibly query for images by specifying both visual features and spatial attributes of the desired images. Most recent content-based image query systems do not provide both types of querying (for example, [102, 112, 105, 3]).

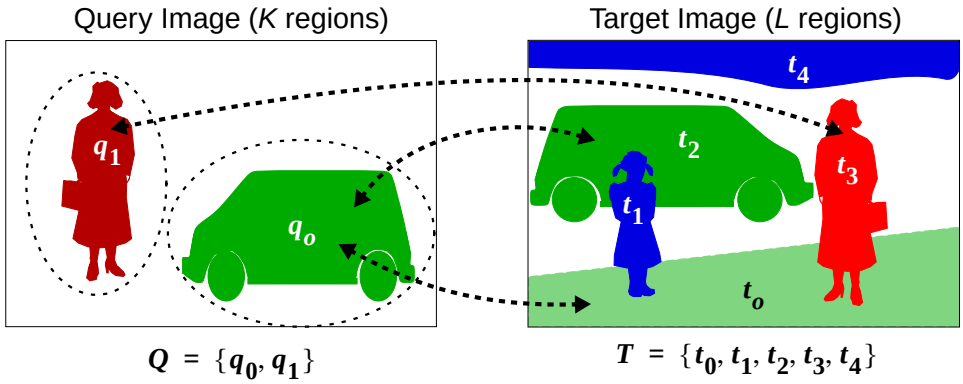


Figure 7-1: Matches are found for each query region q_k. Target images T that contain matches to all query regions are candidate image matches.

The joint spatial and feature image query strategy is summarized in Figure 7-1. The user presents a query image $Q = \{q_0 \ldots q_{K-1}\}$, which consists of K regions. Each of the regions q_k has spatial attributes: $((x, y))$, size $((w, h)$ and `area`) and feature attributes: $(\mathbf{f})$, which are summarized in Table 7.1.

IMID	REGID	f	x	y	w	h	area
q_0	0	$\mathbf{f}_0$	20	50	20	80	1000
q_1	1	$\mathbf{f}_1$	70	65	50	40	1500

Table 7.1: The `REGION` table for query image $Q = \{q_0, q_1\}$.

To compute the query, the system compares the query regions to the regions of the target images T in the database. The system first determines the sets of target regions t_l that sufficiently match each query region q_k. In this process, the system finds matches from each query region to possibly several regions in each target image, as illustrated in Figure 7-1. The target image and its attributes are summarized in Table 7.2.

IMID	REGID	f	x	y	w	h	area
t_0	0	$\mathbf{f}_0$	50	80	100	40	3000
t_1	1	$\mathbf{f}_1$	25	60	10	55	500
t_2	2	$\mathbf{f}_2$	35	45	60	50	2000
t_3	3	$\mathbf{f}_3$	75	55	20	80	1200
t_4	4	$\mathbf{f}_4$	50	10	100	20	1500

Table 7.2: The `REGION` table for target image $T = \{t_0, t_1, t_2, t_3, t_4\}$.

After identifying the candidate target regions, they are joined (using a relational `Join` operation) to identify the best target images and corresponding configurations (pairwise matches of query regions to target regions). Each target image and configuration is then assigned an overall match score to the query image. At this final stage, any specification by the user of relative spatial constraints are checked for each target image using query-time 2-D string projection and 2-D string matching.

In this way, a query specified by the user is translated directly into pruning operations on the region attributes. The image attributes, such as region relative locations and special spatial relations, are resolved only in the final stage of the query. This is because these evaluations have the highest complexity. The pruning performed by the queries on the region attributes reduces the number of candidate images and regions that need to be evaluated at the final stage.

In section 7.2., we present the strategy for matching regions that have attributes of absolute location and size. In section 7.3., we consider the case of matching images with multiple regions that have attributes of size, absolute and relative locations, which includes cases of special relations between regions. Finally, in section 7.4.6 we provide an evaluation and present some examples of querying by joint spatial and color features.

7.2. Region query

In computing matches between regions, we consider the spatial distances and size differences between the query (q) and target (t) regions.

7.2.1 Absolute spatial location

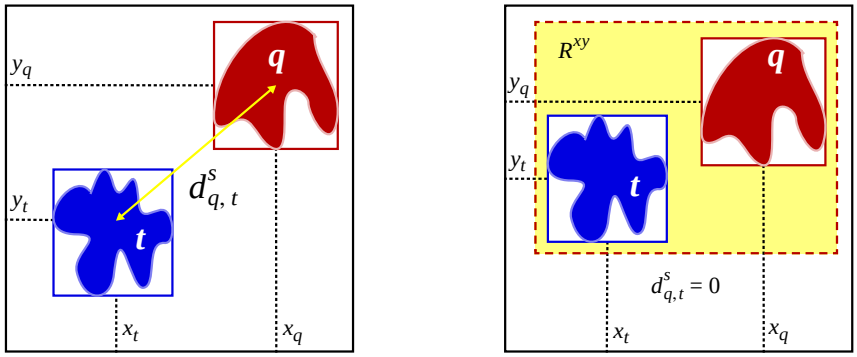
In specifying spatial attributes of the individual regions in the query, we index the centroids and minimum bounding rectangles of the regions.

7.2.1.1 Fixed query location

The spatial location attributes of regions are determined by the spatial centroid of the region (x, y) and the width and height (w, h) of the minimum bounding rectangle (MBR) that encapsulates the region. Both the location and size are useful attributes for discriminating regions. We present techniques for computing the similarity of the locations and sizes of regions and indexing regions by these attributes.

The spatial distance between region centroids is given by the euclidean distance (illustrated in Figure 7-2(a)) as follows

$$d^s_{q,t} = [(x_q - x_t)^2 + (y_q - y_t)^2]^{\frac{1}{2}}. \tag{7.1}$$



(a) (b)

Figure 7-2: Region spatial distance (a) fixed spatial distance $d^s_{q,t}$, (b) bounded spatial distance $d^{s'}_{q,t}$, where R^{xy} determines the valid spatial bounds.

7.2.1.2 Bounded query location

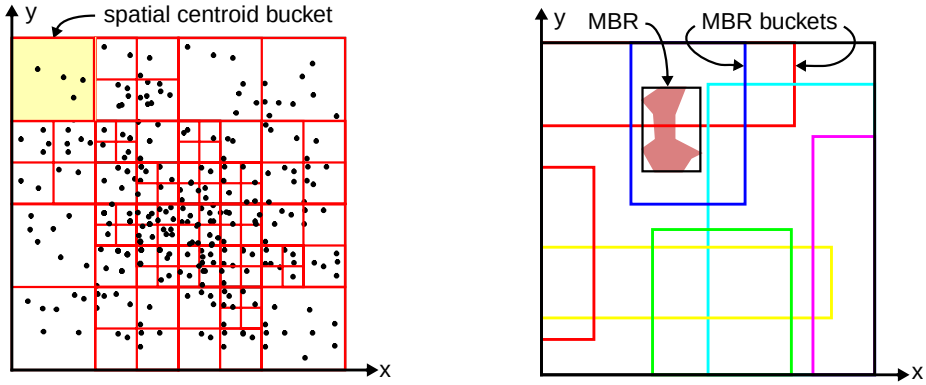
In many situations the user does not care about the exact location of matched regions as long as they fall within a designated area. In this case, the user is given flexibility in designating the spatial bounds for each region in the query within which a target region is assigned a spatial distance of zero. When a target region falls outside of the spatial bounds, the spatial distance is given by the euclidean distance (illustrated in Figure 7-2(b)) as follows

$$d^{s'}_{q,t} = \begin{cases} 0 & \text{if } (x_t, y_t) \in R^{xy} \\ d^s_{q,t} & \text{otherwise.} \end{cases} \tag{7.2}$$

7.2.1.3 Centroid location spatial access – spatial quad-trees

The spatial quad-tree provides an efficient structure for indexing the region centroids. The quad-tree provides quick access to 2-D data points by grouping the regions in buckets [132], as illustrated in Figure 7-3(a). A query for region at location (x_t, y_t) is processed by first traversing the spatial quad-tree to the nearest buckets.

These buckets are searched exhaustively for the points that minimize $d^s_{q,t}$. In the case that the user specifies a bounded spatial query, several buckets are considered such that points within the spatial bounds are all assigned $d^{s'}_{q,t} = 0$.



(a) (b)

Figure 7-3: Spatial indexing techniques, (a) the spatial quad-tree indexes region centroids, and (b) the r-tree indexes region MBRs (rectangles) by using overlapping buckets.

7.2.1.4 Rectangle location spatial access – R-trees

The spatial locations of extracted image regions are not sufficiently represented by centroid locations alone. A 2-dimensional r-tree indexes the regions by their minimum bounding rectangles (MBRs). A MBR is the smallest vertically aligned rectangle that completely encloses the region. In this way, a spatial query may specify a search rectangle and the target regions are found that overlap with it, spatially.

The MBRs of the regions are indexed using the r-tree illustrated in Figure 7-3(b). The r-tree provides a general dynamic structure for indexing $k-D$ rectangles [58], here $k=2$. The r-tree, which consists of a hierarchy of overlapping spatial buckets, is designed to visit only a small number of buckets in the spatial search [58].

7.2.2 Size

Another important perceptual dimension of the regions is their size, specifically in terms of area and spatial extent. The distance in area between two regions q and t, is given by the absolute distance as follows

$$d^a_{q,t} = |\texttt{area}_q - \texttt{area}_t|. \tag{7.3}$$

The width and the height of the MBRs provide for useful comparison of region spatial extents. It is much simpler than shape information, which we do not utilize in the system, and provides excellent grounds for discriminating regions by size. The distance in MBR width (w) and height (h) between two regions q and t, is given by

$$d^m_{q,t} = [(w_q - w_t)^2 + (h_q - h_t)^2]^{\frac{1}{2}}. \tag{7.4}$$

7.2.3 Region query strategies

Integrating these approaches, the overall region query consists of computing individual queries on the feature values, region location, area and spatial extent, as specified by the user.

The single region distance is given by the weighted sum of the feature, location (Eq. 7.1), area (Eq. 7.3) and spatial extent (Eq. 7.4) distances. The user may also assign a relative weighting α_n to each attribute. For example, the user may weight the size parameter more heavily than feature value and location in the query. The overall single region query distance between regions q and t is given by

$$d_{q,t} = \alpha_c d^{set}_{q,t} + \alpha_s d^s_{q,t} + \alpha_a d^a_{q,t} + \alpha_m d^m_{q,t}. \tag{7.5}$$

We now address the query process and present two approaches for computing the region queries.

7.2.3.1 Relational algebra

We introduce some basic concepts from relational databases in order to describe the image query and region query processes. The relational model provides for defining a structure of the image database. Relational algebra is a collection of operations that are used to manipulate the relations [42].

In the relational model, data is organized in tables of values. Each column of a table (or *attribute*) represents a particular dimension of the data (i.e., age or sex). Each row of the table (or *tuple*) represents a collection of related attributes that correspond to one entry in the table (i.e., a person).

More precisely, a relation of size n is denoted by $R(A_0, A_1, \ldots, A_{n-1})$, where the attributes are A_i for $i = \{0 \ldots n-1\}$. An n-tuple t in the relation R is denoted by $t =< v_0, v_1, \ldots, v_{n-1} >$, where v_i is the value corresponding to attribute A_i [42]. The specific operations for querying relational databases include the `Select` (σ), `Project` (π), and `Join` ($\bowtie$) operations.

The `Select` operation returns the tuples from the relation that have attributes with values that satisfy the specified conditions. The syntax of the `Select` operation is $\sigma_{<\text{select condition}>}(<\text{relation}>)$. The select condition is made up of boolean clauses that apply conditions to the tuple values. The `Project` operation returns only specified attributes from the relation. The syntax of the `Project` operation is $\pi_{<\text{attribute list}>}(<\text{relation}>)$.

The `Join` operation is an operation on two relations that combines tuples from both. In general, the `Join` of two relations

$$R(A_0, A_1, \ldots, A_{n-1}) \text{ and } S(B_0, B_1, \ldots, B_{m-1})$$

is denoted by $R \bowtie_{\text{<join condition>}} S$. The result of the `Join` is a relation Q with $n+m$ attributes

$$Q(A_0, A_1, \ldots, A_{n-1}, B_0, B_1, \ldots, B_{m-1}).$$

Q has one tuple for each combination of tuples from R and S that satisfy the `Join` condition.

In particular, the type of `Join` we use is the `Natural Join`. In this case, the operation: $R \bowtie_{\text{<attribute list>}} S$ returns Q where the tuples in Q correspond to the tuples from R and S that have attributes from the attribute list with equal values. In the following discussion, we assume that the `Join` operations are of the form of the `Natural Join`.

7.2.3.2 Relational image database

We structure the image region database using the relation model. This allows for queries to be described using relational algebra and the `Select`, `Project` and `Join` operations. The region relational table (R = `REGION`) is depicted in Table 7.3. Each row or tuple in R gives a region and its attributes. Particular regions are extracted from R using the `Select` operation. For example, all regions from a particular image (say, with `IMID` = 0001) are returned using the operation $\sigma_{\texttt{IMID} = 0001}(R)$.

IMID	REGID	f	x	y	area	w	h
0001	0001	$\mathbf{f}_0$	18	63	430	30	15
0001	0002	$\mathbf{f}_1$	34	45	968	65	32
0002	0001	$\mathbf{f}_2$	76	54	780	53	42
0003	0001	$\mathbf{f}_3$	55	12	654	43	55

Table 7.3: The `REGION` relation with attributes for features $\mathbf{f}$, region centroids (x, y), region sizes `area` and widths and heights (w, h) of the minimum bounding rectangles (MBRs).

7.2.3.3 Parallel attribute query strategy

We first outline the strategy for processing the region attribute queries using a parallel strategy. For example, given the single region query: *find the region that best matches* $Q = \{\mathbf{f}_q, (x_q, y_q), \texttt{area}_q, (w_q, h_q)\}$, the query is processed by first computing individual queries for feature, location, size and spatial extent, as illustrated in Figure 7-4.

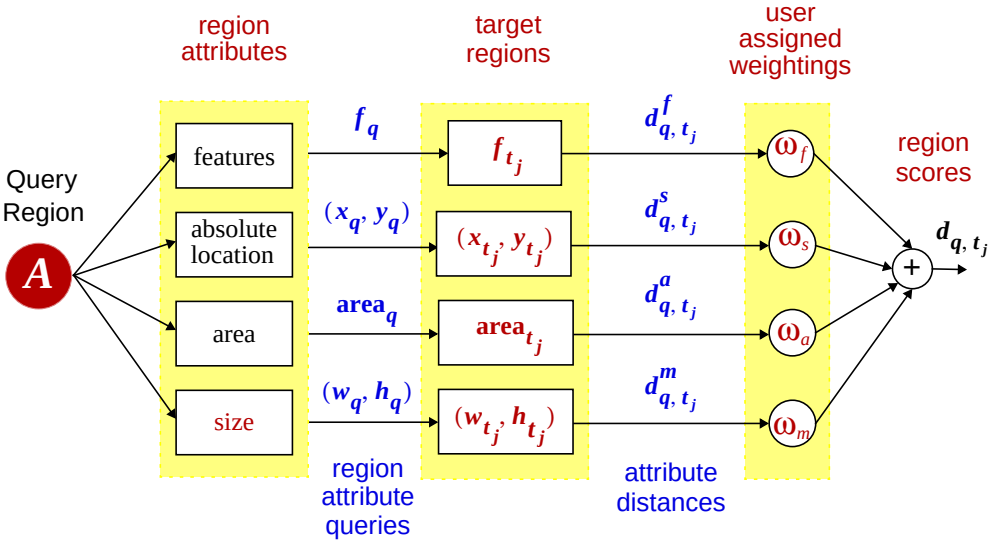


Figure 7-4: Parallel attribute query strategy for a single region query with attributes of feature value, location, area and spatial extent.

The `Join` of the region match lists is then computed to obtain the set of common images; the best match minimizes the total distance. Here, $\mathcal{B}^{xy}$, $\mathcal{A}^{xy}$ and $\mathcal{M}^{wh}$ are the search thresholds for location, area and spatial

extent, respectively, and are set by the system or by the user.

The parallel strategy achieves a performance gain over exhaustive search since it allows for special indexing structures to be utilized for each attribute of the region. For example, the spatial quad-tree is used to index the regions by location.

The parallel region query strategy is carried using the following operations, where the expression,

$$\texttt{Select} * \texttt{from REGION where } < \text{select condition} >$$

is equivalent to the following expression in relational algebra where the $*$ refers to all attributes $\pi_*(\sigma_{<\text{select condition}>}(\texttt{REGION}))$.

$$\begin{array}{lcl}
\texttt{FEATURE} & \leftarrow & \texttt{Feature query}(\mathbf{f}_q) \\
\texttt{LOC} & \leftarrow & \texttt{Select } * \texttt{ from REGION where } (x_t, y_t) \in \mathcal{B}^{xy} \\
\texttt{SIZE} & \leftarrow & \texttt{Select } * \texttt{ from REGION where area}_t \in \mathcal{A}^{xy} \\
\texttt{MBR} & \leftarrow & \texttt{Select } * \texttt{ from REGION where } (w_t, h_t) \in \mathcal{M}^{wh} \\
\texttt{CAND} & \leftarrow & \texttt{FEATURE} \bowtie_{\text{IMID}} \texttt{LOC} \bowtie_{\text{IMID}} \texttt{SIZE} \bowtie_{\text{IMID}} \texttt{MBR} \\
\texttt{REGIONMATCH} & \leftarrow & \min_{d_{q,t}}(\texttt{CAND}), \text{ (using Eq. 7.5).}
\end{array}$$

7.2.3.4 Pipeline attribute query strategy

We present an alternate strategy for processing the region attribute queries for a single region using a pipeline query strategy. For example, given the single region query: *find the region that best matches* $Q = \{\mathbf{f}_q, (x_q, y_q), \texttt{area}_q, (w_q, h_q)\}$, the query is processed by first computing the individual query on the feature attributes. This output is then filtered by location, then by size and, finally, by spatial extent, as illustrated in Figure 7-5.

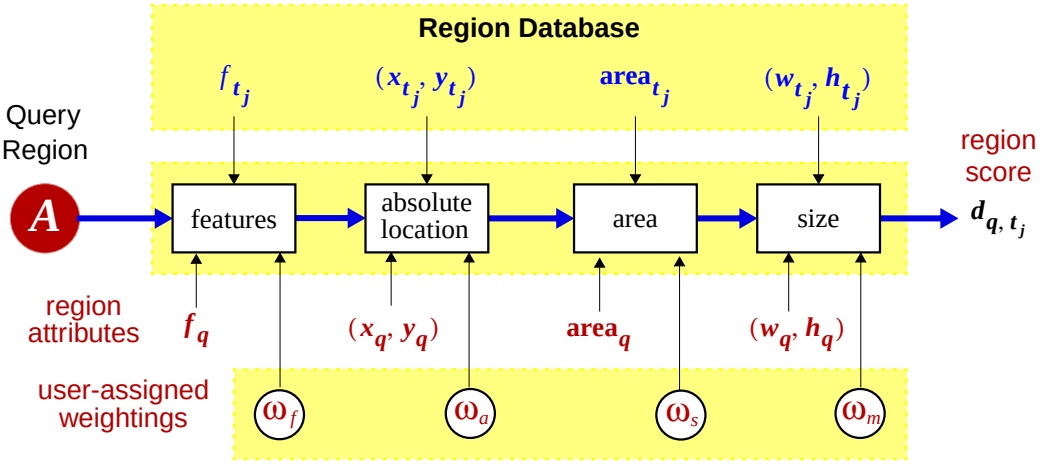


Figure 7-5: Pipeline attribute query strategy for a single region query with attributes of feature value, location, area and spatial extent.

The pipeline strategy avoids the computation of the attribute `Join` required in the parallel strategy. However, while a special indexing structure may still be used on the feature values, the pipeline strategy cannot utilize special indexes for the other attributes that follow in the pipeline.

$$\begin{array}{lcl}
\texttt{FEATURE} & \leftarrow & \texttt{Feature query}(\mathbf{f}_q) \\
\texttt{LOC} & \leftarrow & \texttt{Select } * \texttt{ from FEATURE where } (x_t, y_t) \in \mathcal{B}^{xy} \\
\texttt{SIZE} & \leftarrow & \texttt{Select } * \texttt{ from LOC where area}_t \in \mathcal{A}^{xy} \\
\texttt{CAND} & \leftarrow & \texttt{Select } * \texttt{ from SIZE where } (w_t, h_t) \in \mathcal{M}^{wh} \\
\texttt{REGIONMATCH} & \leftarrow & \min_{d_{q,t}}(\texttt{CAND}), \text{ (using Eq. 7.5).}
\end{array}$$

7.3. Multiple regions query

The overall image query strategy consists of joining the queries on the individual regions in the query image. The `Join`, which consists of the `Natural Join` of the `REGIONMATCH` tables, identifies the candidate target images. For these images, the image match score is computed by adding the weighted region scores.

In the final stage, the relative spatial locations that may have been specified in the query are evaluated using query-time 2-D string projection. A 2-D string comparison determines whether candidate target images

satisfy the constraints of the relative region placement. The image match process is illustrated in Figure 7-6. We note that by specifying multiple regions in the query without any spatial relations between regions, the complexity increases only linearly with the number of query regions.

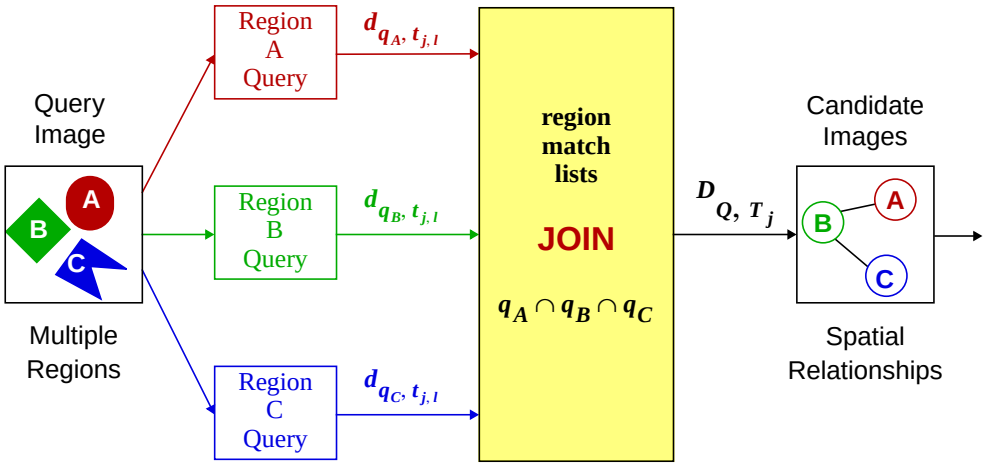


Figure 7-6: Overall strategy for computing image matches by joining individual regions queries.

7.3.1 Multiple regions query strategy – absolute locations

For each region in the query positioned by absolute location, the query strategy outlined for single region query (Section 7.2.3) is carried out, without computing the final minimization. These lists are joined and the best image match minimizes the combined region distance. For example, given the multiple region query: *find the image having three regions that best matches* $Q = \{Q^A, Q^B, Q^C\}$, where $Q^i = \{\mathbf{f}_q^i, (x_q^i, y_q^i), \texttt{area}_q^i, (w_q^i, h_q^i)\}$ gives the query parameters for query region i, the matches are found as follows:

$$
\begin{array}{lcl}
\texttt{REG}_A & \leftarrow & \texttt{Single Region Query}(Q^A) \\
\texttt{REG}_B & \leftarrow & \texttt{Single Region Query}(Q^B) \\
\texttt{REG}_C & \leftarrow & \texttt{Single Region Query}(Q^C) \\
\texttt{CAND} & \leftarrow & \texttt{REG}_A \bowtie_{\text{IMID}} \texttt{REG}_B \bowtie_{\text{IMID}} \texttt{REG}_C \\
\texttt{IMAGEMATCH} & \leftarrow & \min_{\beta_0 d_{Q_A,T} + \beta_1 d_{Q_B,T} + \beta_2 d_{Q_C,T}} (\texttt{CAND}).
\end{array}
$$

The `REGIONMATCH` relations are `Joined` to obtain the `IMAGEMATCH` relation. The best image matches from `IMAGEMATCH` minimize the weighted sum of the region distances between the query and target image. An example of a query that specifies four regions with absolute locations is illustrated in Figure 7-7. The query image is given in the upper left. The matches are listed from left to right, and top to bottom in order of minimum distance to the query image.

7.3.2 Region relative location

Querying by absolute locations cannot be easily extended to include relative locations of regions. For example, for a target with L regions and a query image with K regions, there are $L!/(L-K)!K!$ possible matches, each of which has a different distance score. For example, $L = 20$ and $K = 3$ requires $1,140$ comparisons. To circumvent this exhaustive search, we utilize a convenient representation of image spatial relationships and their comparisons based upon 2-D strings [22].

The 2-D string represents spatial relationships as illustrated in Figure 7-8(a). The 2-D string is a symbolic projection of the image along the x and y directions [20]. For example, the 2-D string for the image in Figure 7-8(a) is given as $(t_0 t_1 < t_2 < t_7 < t_3 < t_6 < t_4 < t_5, t_0 < t_5 t_7 < t_6 < t_2 < t_3 t_1 < t_4)$, where the symbol '$<$' denotes the left-right or bottom-top relation.

Using 2-D strings, the match complexity is reduced to approximately $O(L^3/K)$ per target image. Since we evaluate spatial relationships only at the final stage of the query, the complexity is further reduced to $O(L'^3/K)$, where L' is the number of candidate regions in a target image, and $L' < L$ and $L' \simeq K$. Using the region `Join` strategy in Section 7.3., we have $L' = K$, which gives a complexity of $O(K^2)$.

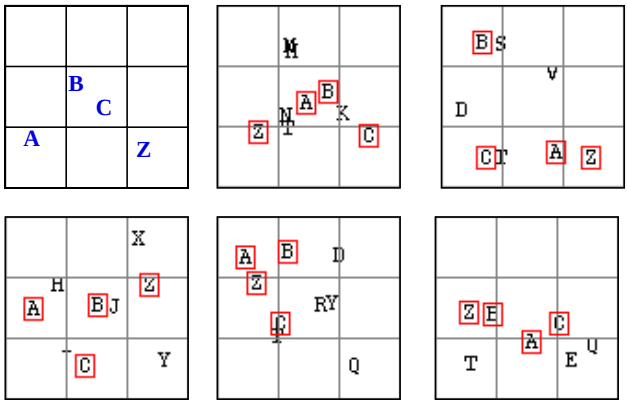
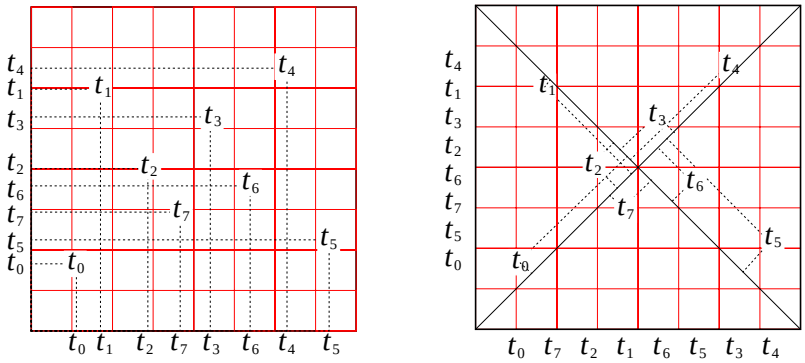


Figure 7-7: Symbolic image retrieval – absolute spatial query matches.



(a) (b)

Figure 7-8: 2-D string representation of spatial relationships (a) un-rotated 2-D string, (b) rotated projection.

7.3.3 Spatial indexing

In order to incorporate relative spatial query into the feature query system, the 2-D strings are generated at the final stage of the query.

7.3.3.1 Query-time 2-D string projection

The 2-D string projection (illustrated in Figure 7-9) is accomplished as follows: given a `REGION` or `IMAGEMATCH` table, the centroid (x, y) attributes are first quantized. The quantization, specified by step size Δ, is determined by the size of the grid, as depicted in Figure 7-8. From the quantized relation, two new relations are obtained: `SX` and `SY`. For example, `SX` is obtained by the `Select` operation on attributes `IMID`, `SYMB`, and `X` and sorting by `IMID` and `X`. The `SYMB` are then concatenated in sorted order into a 'comma delimited' string. The process is repeated for the `SY` table and the `Y` attribute. Finally, `SX` and `SY` are `Joined` to produce the 2-D string relation `2DS`.

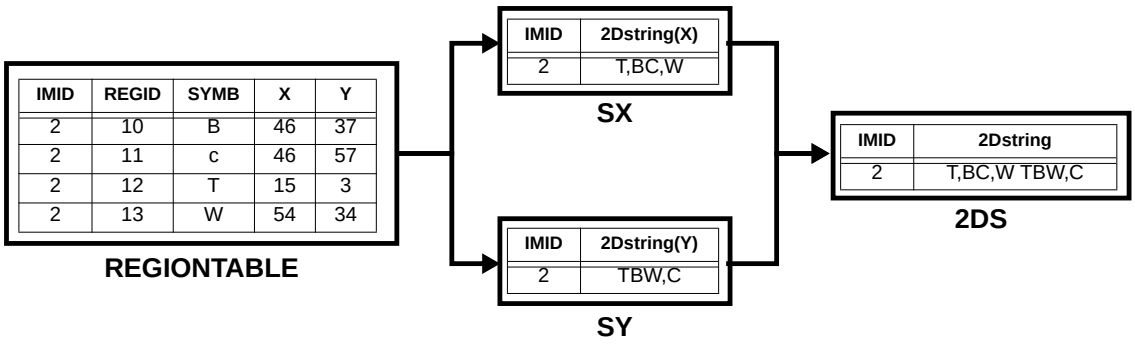


Figure 7-9: 2-D String projection.

The following algorithm summarizes the 2-D string projection process.

Algorithm 1 2-D String Projection

$$
\begin{array}{lcl}
\mathrm{Q} & \leftarrow & \mathrm{quantize}_{\mathrm{X,\ Y}}(\texttt{REGIONTABLE}, \Delta) \\
SX & \leftarrow & \mathrm{sort}_{\mathrm{ID,\ SYMB,\ X}}(\sigma_{\mathrm{ID,\ X}}(\mathrm{Q})) \\
SY & \leftarrow & \mathrm{sort}_{\mathrm{ID,\ SYMB,\ Y}}(\sigma_{\mathrm{ID,\ Y}}(\mathrm{Q})) \\
2DS & \leftarrow & SX \bowtie_{\mathrm{ID}} SY
\end{array}
$$

7.3.3.2 2-D String compare

Given the 2-D string projections from the query image Q and target images T, the spatial relation constraints in Q are evaluated in the T's by a 2-D string comparison. The 2-D strings comparison is carried out using the *2dcompare* function:

Algorithm 2 2-D String Compare *2dcompare(Q, T)*

$$2dcompare(Q, T) \quad \leftarrow \quad compare(Q_X, T_X) \texttt{ and } compare(Q_Y, T_Y).$$

The 2-D string comparison is separated into individual `X` and `Y` 1-D string comparisons, which are evaluated using the *compare* function as follows

Algorithm 3 Compare *compare(Q, T). The pseudo-code for the compare function is given in Figure 7-10, where Q is the query string, T is the target string, NUM_Q is the number of buckets in the query string, and NUM_T is the number of buckets in the target string.*

The *contains* function implements a search for a query symbol within one bucket of the target string.

An example of a query that specifies two regions with relative locations is illustrated in Figure 7-11. The query image is given in the upper left. The matches satisfy the spatial constraints specified in the query image, namely, that region A is above and the right of region B.

```
let nq = 0
let nt = 0
while nq < NUMQ and nt < NUMT do
   if contains(T[nt],Q[nq]) then
      nq = nq + 1
   endif
   nt = nt + 1
end
if nq < NUMQ then
   return "NO MATCH"
else
   return "MATCH"
endif
```

(a)

```
function contains(T[nt], Q[nq])
   while (i < size(Q[nq]) do
      if (substr(T[nt],Q[nq][i])
         then return FALSE;
      endif
    i = i + 1;
   end
   return TRUE;
```

(b)

Figure 7-10: 2-D string compare pseudo-code.

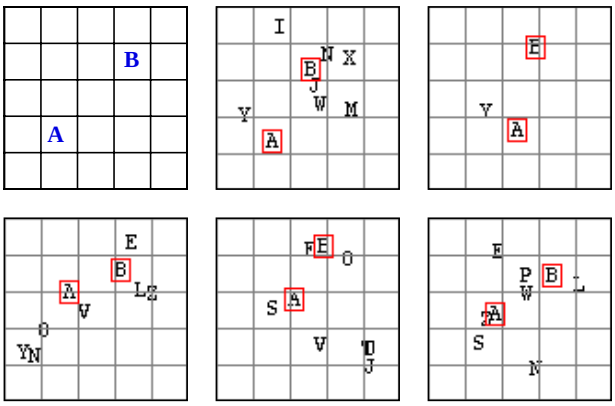


Figure 7-11: Symbolic image retrieval – relative spatial query matches.

7.3.4 Special spatial relations

We define several spatial relations that are of particular interest, such as region adjacency, nearness, overlap and surround. These examples of spatial reasoning are inferred from the 2-D string [20]. Scale invariance is also provided in the 2-D string, but rotation invariance is not. We introduce a simple extension of the 2-D string projection and comparison operations that gives some rotation invariance.

7.3.4.1 Adjacency, nearness, overlap and surround

The adjacency of regions is resolved from the 2-D string by determining whether or not other regions exist between two candidate regions. The nearness of regions is resolved using adjacency in the 2-D string, but since the 2-D string provides no metric information, it requires an additional computation of the region distance.

Since the 2-D string captures the orthogonal relations of the regions in the image, the conditions of overlap and surround can also be determined [20]. Overlapping regions require not only the determination of nearness but also require the evaluation of the overlap of the MBRs of the regions. The case of surround is determined by unifying sub-goal queries that evaluate regions to the north, south, east and west of the surrounded-by region.

An example of a "nearness" query that specifies three regions with relative locations is illustrated in Figure 7-12. The query image is given in the upper left. By defining a coarse grid and placing the three regions A, B and C in the same bucket, images are retrieved that also have the regions in the same bucket.

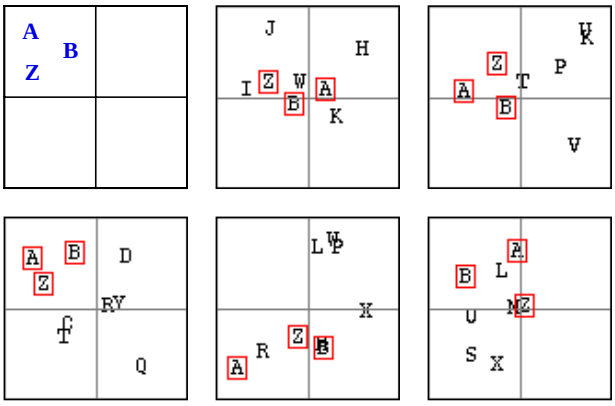


Figure 7-12: Symbolic image retrieval – "nearness" relative spatial query matches.

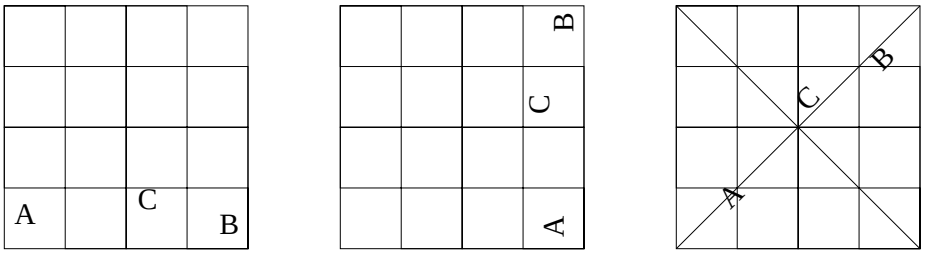
7.3.4.2 Spatial invariance

Under certain conditions the user may desire invariance in scaling and/or rotation. The 2-D string provides scale invariance since a change in scale does not change the order of symbols in the string. However, rotation invariance is not provided in the 2-D string. Therefore, we approximate rotation invariance around the image center point by providing two projections of each image, one normal projection, and another based upon a rotation by 45 degrees.

In this case, the 45° rotated 2-D string is extracted by the projecting onto the diagonals of the image as illustrated in Figure 7-8(b). Image rotation by 90° is provided by swapping the x and y projections as illustrated in Figure 7-13(a) and (b). Rotation by 45° is detected in the rotated 2-D string as illustrated in Figure 7-13(a) and (c). Other arbitrary rotations are approximated by either the 0° 2-D string or the 45° or 90° rotated 2-D string.

7.3.5 Multiple regions query strategy – relative locations

To summarize the multiple region query strategy, the resolution of the relative spatial locations is performed in the final stage. For the regions positioned in the query by absolute locations, the location queries from Section 7.3.1 are performed. For the regions positioned in the query by relative locations, queries on all



(a) 0° (b) 90° (c) 45°

Figure 7-13: 2-D strings rotation invariance (a) 0° ($A < C < B, ABC$), (b) 90° ($ABC, A < C < B$), (c) 45° ($A < C < B, ABC$) with projection onto the rotated axis.

attributes except location (x, y) are performed. Then, the `Join` of the `REGIONMATCH` tables is performed to identify and score the candidate target images.

For each candidate image, the 2-D string is generated by projecting from the centroids of the regions and is compared to the 2-D string of the query image. This final 2-D string comparison operation either validates the target image or rejects it. Therefore, after the final stage, all false alarms from the earlier stages are removed.

An example of a query that specifies two regions with both absolute and relative locations is illustrated in Figure 7-14. The query image is given in the upper left. The matches satisfy the spatial constraints specified in the query image, namely, that region A is above and the right of region B. We now present some example multiple region spatial queries using symbolic images, and provide some evaluations.

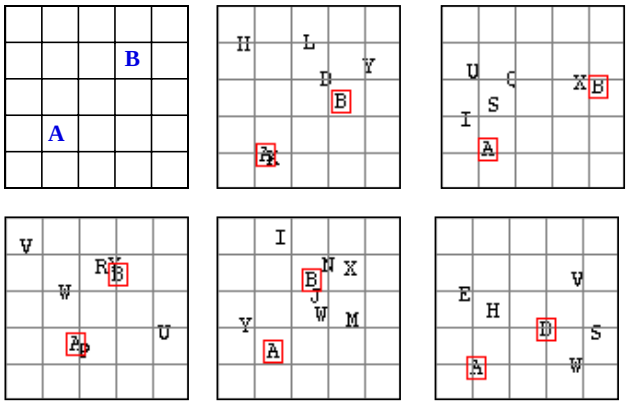


Figure 7-14: Symbolic image retrieval – absolute spatial query matches.

7.3.6 Symbolic image retrieval

The spatial queries are formulated graphically using the interface tools illustrated in Figure 7-15 [142]. The user sketches regions, positions them on the query grid and assigns them attributes of color, size and absolute location. The user may also assign boundaries for location and size.

7.3.7 Relative spatial query efficiency

Figure 7-16 reports on the efficiency of the 2-D string approach in computing relative spatial queries. Figure 7-16(a) illustrates that the query response time using 2-D strings is approximately linear with the size of the image database. This follows since we do not utilize an index structure for the 2-D strings.

The `IMAGEMATCH` table is searched exhaustively using the 2-D compare function. Figure 7-16(b) illustrates that the observed query response time is approximately constant with the number of query symbols.

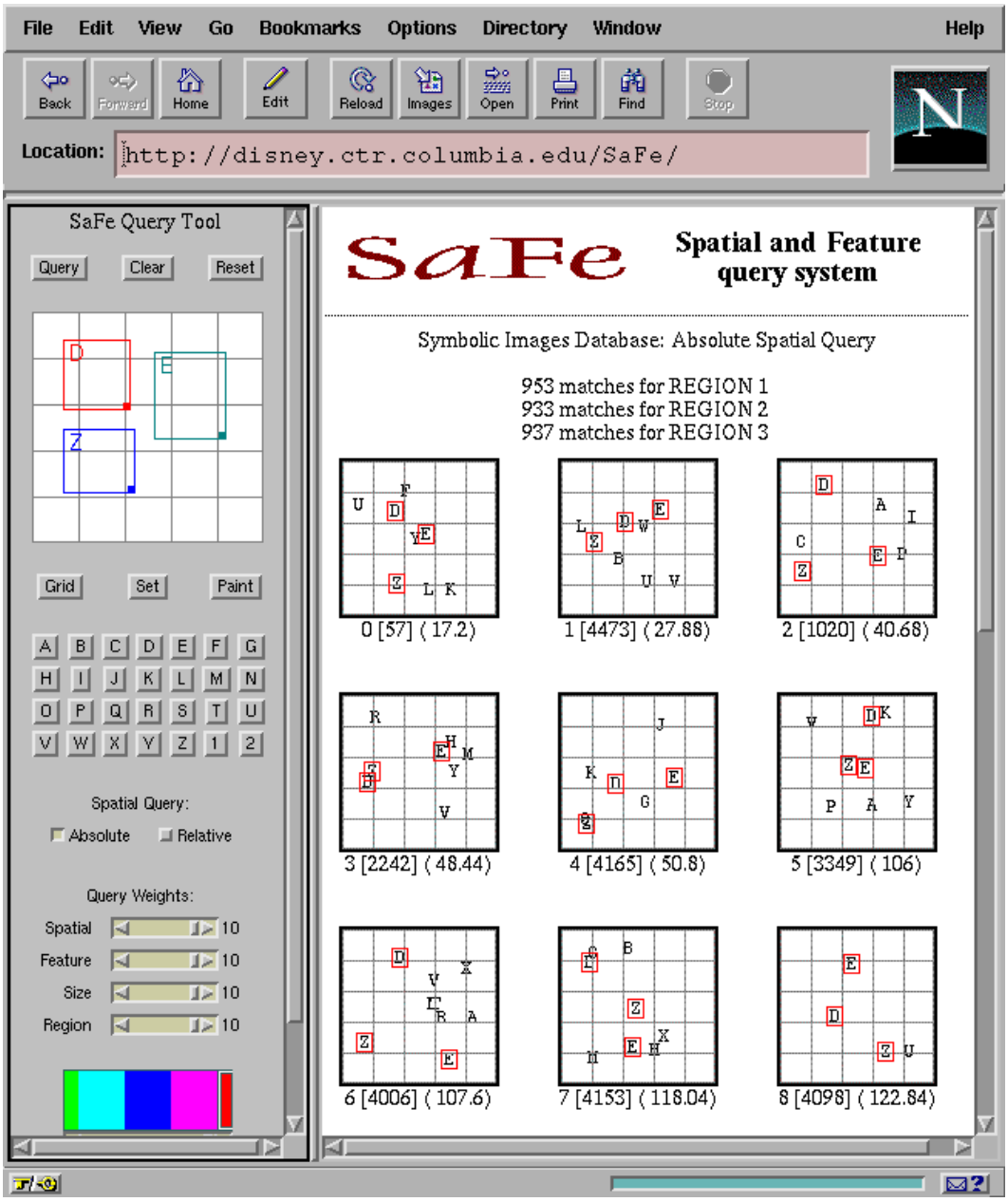


Figure 7-15: SaFe user interface with returned symbolic images.

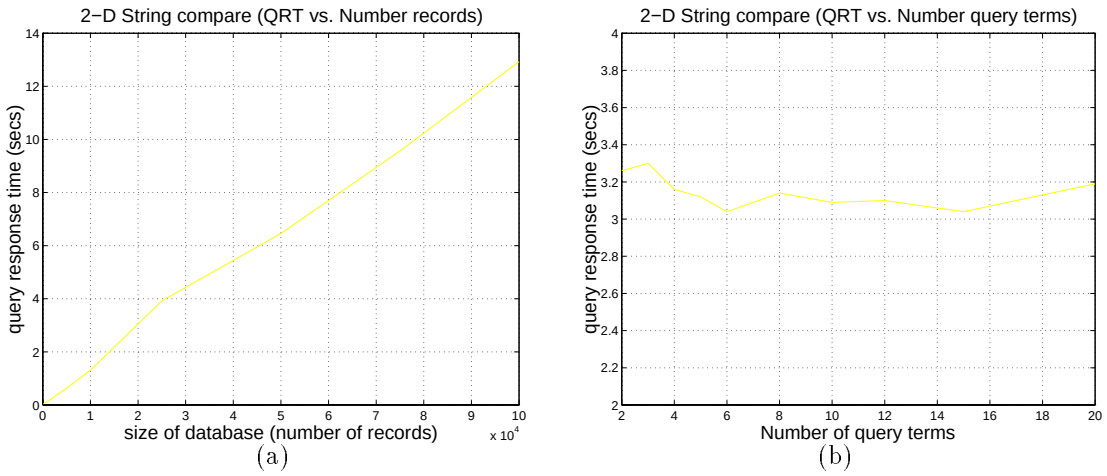


Figure 7-16: Relative spatial query (a) query response time *vs.* image database size, (b) query response time *vs.* number of query symbols.

7.4. Feature query

The power of the spatial and feature query system comes from its capability to integrate content-based searching with spatial indexing. The process described above provides a framework for computing these queries. We now examine one specific case of spatial and feature querying in which the feature of interest is color. Query-by-color is one popular method for color image retrieval [3, 45, 105]. By extending query-by-color to the spatial and color feature paradigm, we create a more powerful system for retrieving color images. We now describe the process of representing and comparing color features, extracting color regions and computing color feature queries.

7.4.1 Histogram quadratic distance

Color histograms represent the color features of images (see Chapter 2). The QBIC project uses the histogram quadratic distance metric for matching images [102]. It measures the weighted similarity between histograms which provides more desirable results than "like-bin" only comparisons. The quadratic distance between histograms $\mathbf{h}_q$ and $\mathbf{h}_t$ is given by

$$d^{hist}_{q,t} = (\mathbf{h}_q - \mathbf{h}_t)^t \mathbf{A} (\mathbf{h}_q - \mathbf{h}_t), \tag{7.6}$$

where $\mathbf{A} = [a_{i,j}]$ and $a_{i,j}$ denotes the similarity between colors with indices i and j.

7.4.2 Color sets

Color sets provide a compact alternative to color histograms for representing color information. Their utilization stems from the conjecture that salient regions have not more than a few, equally prominent colors. The following paragraphs define color sets and explain their relationship to color histograms.

7.4.2.1 Color set distance

We use a modification of the color histogram quadratic distance equation (from Eq. 7.6) to measure the distance between color sets. The quadratic distance between two color sets $\mathbf{c}_q$ and $\mathbf{c}_t$ is given by

$$d^{set}_{q,t} = (\mathbf{c}_q - \mathbf{c}_t)^t \mathbf{A} (\mathbf{c}_q - \mathbf{c}_t). \tag{7.7}$$

Considering the binary nature of the color sets, the computational complexity of the quadratic distance function may be reduced. We decompose the color set quadratic formula to provide for more efficient computation and indexing. By defining $\mu_q = \mathbf{c}_q^t \mathbf{A} \mathbf{c}_q$, $\mu_t = \mathbf{c}_t^t \mathbf{A} \mathbf{c}_t$ and $\mathbf{r}_t = \mathbf{A}\mathbf{c}_t$, the color set quadratic distance is given as

$$d^{set}_{q,t} = \mu_q + \mu_t - 2\mathbf{c}_q^t \mathbf{r}_t. \tag{7.8}$$

Since $\mathbf{c}_q$ is a binary vector, we have

$$d^{set}_{q,t} - \mu_q = \mu_t - 2 \sum_{\forall m \text{ where } \mathbf{c}_{q[m]=1}} r_t[m]. \tag{7.9}$$

That is, any query for the most similar color set to $\mathbf{c}_q$ is easily processed by accessing individually μ_t and $r_t[m]$'s, where $m \in 0 \ldots M-1$, see Table 7.4. As such, μ_t and $\mathbf{r}_t[m]$'s are precomputed, stored and indexed individually. Notice also that μ_q is a constant of the query. The closest color set, $\mathbf{c}_t$, to $\mathbf{c}_q$ is the one that minimizes Eq 7.9.

IMID	REGID	μ^*	$r[0]^*$	$r[1]^*$	...	$r[M-1]^*$

Table 7.4: The COLORSET relation with attributes for the decomposed quadratic distance equation parameters μ_t and $r_t[m]$'s. Denotation by * indicates that a secondary index is built on the attribute in order to allow range queries to be performed on that attribute.

7.4.3 Color region extraction

In order to extract color regions, we use a color set back-projection technique. We briefly describe the technique here. The more detailed presentation is given in Chapter 5. The back-projection process requires several stages: color set selection, back-projection onto the image, thresholding and labeling. Candidate color sets are selected first with one color, then with two colors, etc., until the salient regions are extracted.

The back-projection of a color set is accomplished as follows: given image $I[x, y]$ and color set $\mathbf{c}$, let k be the index of the color at image point $I[x, y]$, then generate image $B[x, y]$ by

$$B[x, y] = c[k], \quad \text{(binary back-projection)}. \tag{7.10}$$

That is, $B[x, y]$ depicts the back-projection of color set $\mathbf{c}$. In order to accommodate color similarity in the back-projection process, a correlated back-projection image is generated by

$$B[x, y] = \max_{j=0 \ldots M-1} (A_{j,k} c[j]), \quad \text{(correl. back-projection)}, \tag{7.11}$$

where $A_{j,k}$ measures the similarity of colors j and k, which will be discussed in the next section. After back-projecting the model color set, image $B[x, y]$ is filtered and analyzed to reveal spatially localized color regions.

7.4.3.1 Color region information

Information about the regions such as the color set used for back-projection, the spatial location and size are added to the REGION relation (see Table 7.3) and are subsequently used for queries as explained in the next sections. While color set selection and back-projection provides one automated technique for extracting salient color regions from the images, other methods can easily be incorporated into our system. For example, manually extracted regions and their features can also be added to the REGION relation.

7.4.4 Color set query strategy

We now define the strategy for processing the color set queries, which is an important building block in the overall color image query process. The color set query compares only the color content of regions or images. The spatial queries were considered in the previous sections. Given query $Q = \{\mathbf{c}_q\}$, the best match to Q is target $T_j = \{\mathbf{c}_{t_j}\}$, where,

$$\begin{aligned} j &= \text{argmin}_j(d^{set}_{q,t_j}) \\ &= \text{argmin}_j(\mu_{t_j} - 2 \textstyle\sum_{\forall m \text{ where } \mathbf{c}_{q[m]=1}} r_{t_j}[m]). \end{aligned}$$

A straightforward way to process the query is to compute d^{set}_{q,t_j} exhaustively for all j using Eq. 7.9. In this case, the computation of d^{set}_{q,t_j} requires $M' + 1$ additions per target and no multiplications, where M' is the

number of non-zero colors in $\mathbf{c}_q$. This provides for drastic improvement over the original histogram quadratic form.

The complexity of this approach, where M = size of color set (histogram) and N = number of images in the database, is $O((M'+1)N)$, where $M' << M$ gives the number of non-zero colors in the query color set. Compare this to a naive computation of quadratic histogram distance, which is $O(M^2N)$. In QBIC, the computation is reduced by the technique in [59] to $O(M^2C + M''N)$, where typically $M'' = 3$ and $C << N$. We note that the technique in [59] may be combined with the color set technique to further reduce complexity.

Given the query color set: *find the best match to color set* $\mathbf{c}_q$, where $d^{set}_{q,t} \leq \tau^c$ defines the maximum tolerance for color set distance, the following range queries produce the best match:

$$
\begin{array}{lll}
\texttt{MU} & \leftarrow & \texttt{Select} * \texttt{ from COLORSET where } \mu_t \leq \tau^c - \mu_q \\
\texttt{COL}_m & \leftarrow & \texttt{Select } r_t[m] \texttt{ from COLORSET where } r_t[m] \geq \frac{\mu_q + \mu_t - \tau^c}{2}, \\
 & & \forall m \texttt{ where } c_q[m] = 1 \\
\texttt{CAND} & \leftarrow & \texttt{MU} \bowtie_{\text{IMID}} \texttt{COL}_i \bowtie_{\text{IMID}} \texttt{COL}_j \bowtie_{\text{IMID}} \cdots \bowtie_{\text{IMID}} \texttt{COL}_k \\
\texttt{COLORMATCH} & \leftarrow & \min_{d^{set}_{q,t}}(\texttt{CAND}), \quad (Eq.\ 7.9).
\end{array}
$$

As this query strategy illustrates, color region matching is accomplished by performing several range queries on the query color set's colors, taking the `Join` of these lists and minimizing the sum of attributes in the joined list. The best match minimizes the color set distance.

Since this strategy merely provides efficient indexing of the terms in the decomposed quadratic color set distance equation, the query strategy provides for no false dismissals. The final candidate image list has false alarms which are removed through the final minimization which requires only additions in the computation of $d^{set}_{q,t}$. Using a similar indexing strategy for retrieving images using color histograms in a database of 750,000 images, the computation of the 60 best matches takes less than two seconds [148].

7.4.5 Synthetic color region image retrieval

To test the spatial and color query system, we generated 500 synthetic images by selecting, manipulating and compositing color regions into synthetic images. Some examples are illustrated in Figure 7-17 and Figure 7-18. While the synthetic images appear quite different from real images, the composition of regions is still difficult to decipher automatically by computer. For example, in the retrieved images in Figure 7-17, the original composited regions cannot be extracted exactly because of occlusion and aggregation of the regions.

The region library consists of twelve elementary shapes, illustrated at the top of Figure 7-19. To construct each synthetic image, shapes were selected at random from the library and were randomly rotated, scaled, colored and composited, as illustrated in Figure 7-19. The control factors, such as color, size, and location of regions, establish a ground truth database.

The synthetic image query experiment was conducted as follows: 100 synthetic query images were generated which have from one to three randomly selected regions in each; 500 target images were generated as described above. For each query image, all of the target images were assigned a similarity score using the ground truth database and using an exhaustive comparison and distance minimization over all query and target regions (Query GT).

The target images from each Query GT were sorted by closest distance to the query image. The target images were assigned a relevance to each query based upon rank in the sorted list as follows:

- if rank = 1 to 5 then relevance = 1,
- if rank = 6 to 10 then relevance = 0.5,
- if rank = 11 to 500 then relevance = 0.

Using the target image relevances obtained from Query GT, the queries were next evaluated using several methods and compared to the Query GT.

1. In Query Q1, the region indexing and distance computation strategy outlined here was carried out on the ground truth database.
2. In Query Q2, the same query strategy was carried out on a region database that was generated automatically from the target images using color set back-projection.
3. In Query Q3, the combined color histogram of the query regions was matched to the combined region color histogram of each target image as the basis for image retrieval.

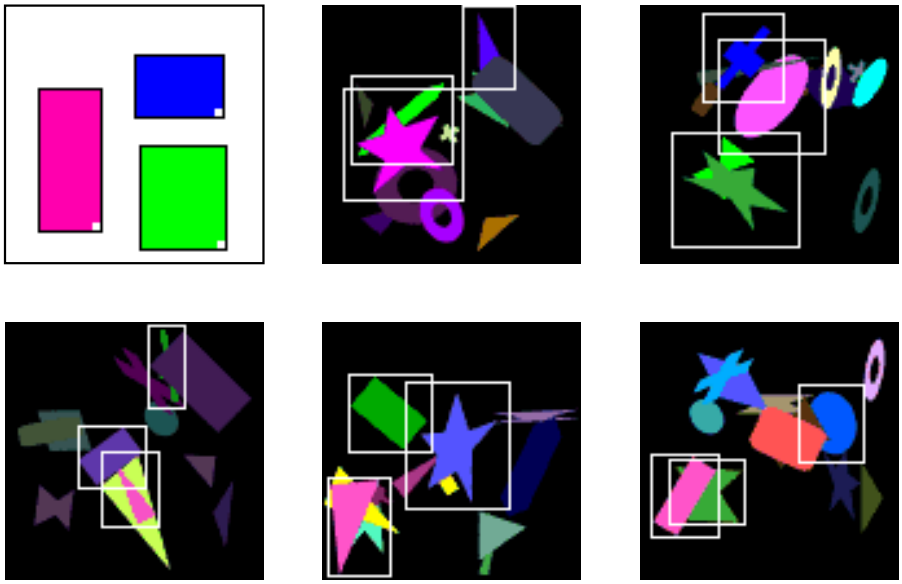


Figure 7-17: Example of the absolute location synthetic image queries: query image is top left, retrieved images from top left to bottom right in order of best match.

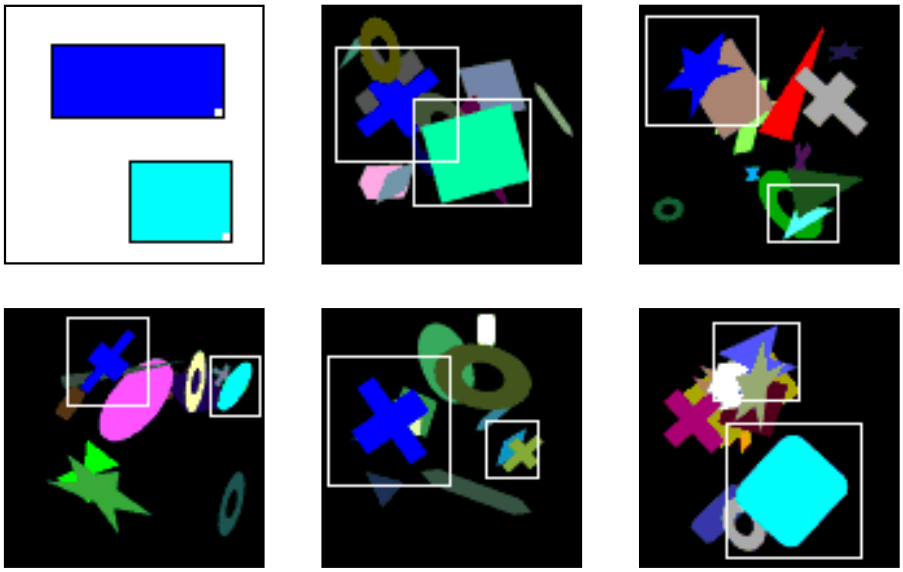


Figure 7-18: Example of the relative location synthetic image queries: query image is top left, retrieved images from top left to bottom right in order of best match.

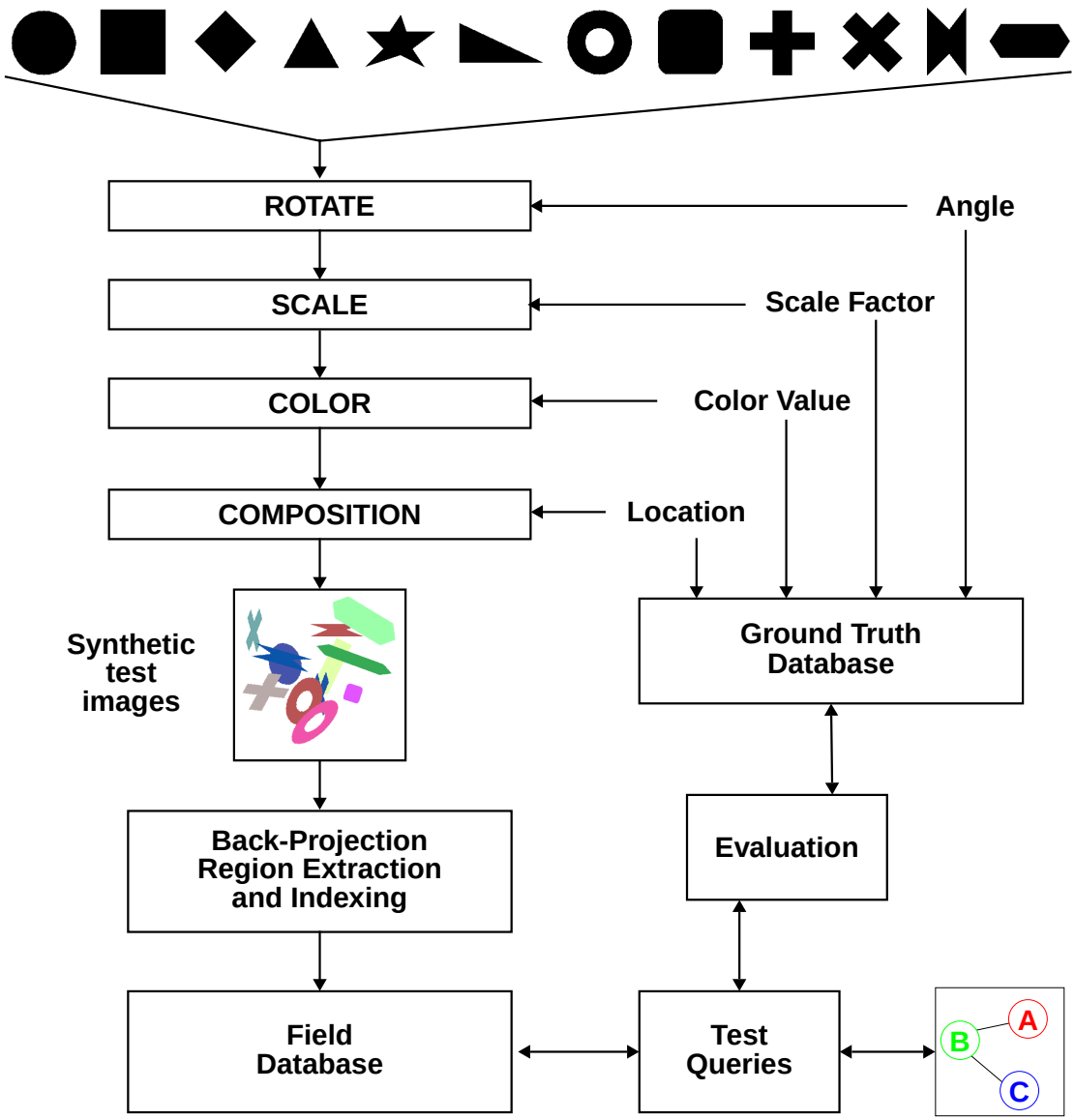


Figure 7-19: Generation of synthetic images and ground truth database and process for evaluating effectiveness of joint spatial and color queries.

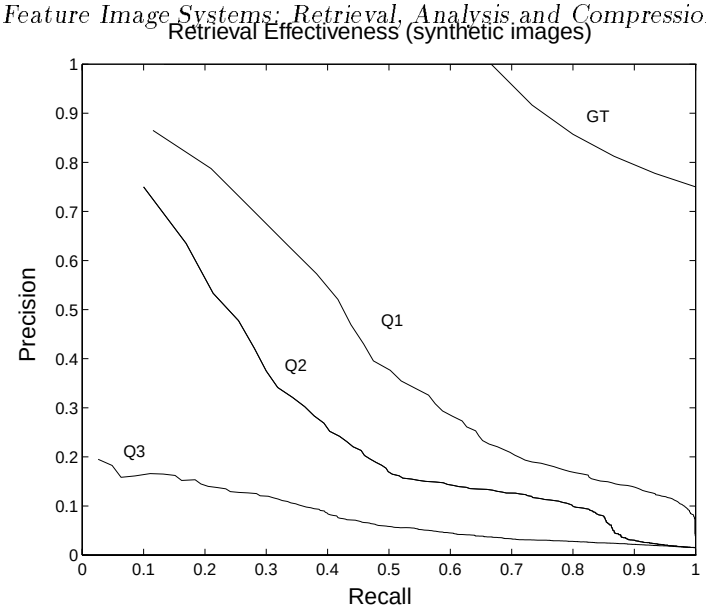


Figure 7-20: Average retrieval effectiveness of 100 randomly generated queries on database of synthetic images, with the four query methods defined as above.

We see in Figure 7-20 that the spatial and color query strategy (Query Q2) performs much better than color histograms (Query Q3) in retrieving image matches. We also see that Query Q1 performs better than Query Q2. The difference represents the loss of information in the process of automated region extraction through color set back-projection. Furthermore, the drop in retrieval effectiveness in Queries Q1 and Q2 from the ground truth Query GT results from the spatial and color indexing strategies.

In computing the similarity between the query and target images in Query GT, all regions in the target are considered. In this way, some target images are identified as matches even when only two out of three regions are close matches. These configurations are considered only because Query GT conducts an exhaustive search on all target regions. In the indexing strategy outlined here, candidate target images are required to possess regions which are all sufficiently close to the query regions. This restriction, which allows the gain in retrieval efficiency over exhaustive search, explains the retrieval effectiveness drop compared to Query GT.

7.4.6 Color photographic image retrieval

The joint spatial and color queries are formulated graphically, as illustrated in Figure 7-21 [152]. The user sketches regions, positions them on the query grid and assigns them attributes of color, size and absolute location. The user may also assign boundaries for location and size. We illustrate the power and flexibility of the system over non-spatial techniques.

7.4.6.1 Query 1: Sunsets

The goal of this query is to retrieve images depicting sunsets. Prior to the trials, the $3,100$ images were inspected, and each was assigned a relevance. The query results are shown in Figure 7-22. The results show that the integrated spatial and color query system gives better retrieval effectiveness than color histogram and color set-based query methods.

Using the SaFe interface (from Figure 7-21), the SaFe sunset query (denoted by $\mathcal{A}$) specifies two regions (outer is orange and inner is yellow) and their spatial layout (top left of Figures 7-22). The best matches (illustrated on top from left to right) have a similar arrangement of similarly colored regions.

For the color histogram (denoted by $\mathcal{B}$) and color set (denoted by $\mathcal{C}$) queries, the best match sunset image from $\mathcal{A}$ is used as the seed image. Query $\mathcal{B}$ is computed using the histogram quadratic distance metric (Eq 7.6). Query $\mathcal{C}$ is computed using the binary set quadratic distance metric (Eq 7.7). The best matches are illustrated from left to right. We see that the global color queries give the user little control in specifying the query and more readily returns images that are not desired. The plot in Figure 7-22 shows that the retrieval effectiveness for $\mathcal{B}$ and $\mathcal{C}$ is much worse that the SaFe query $\mathcal{A}$.

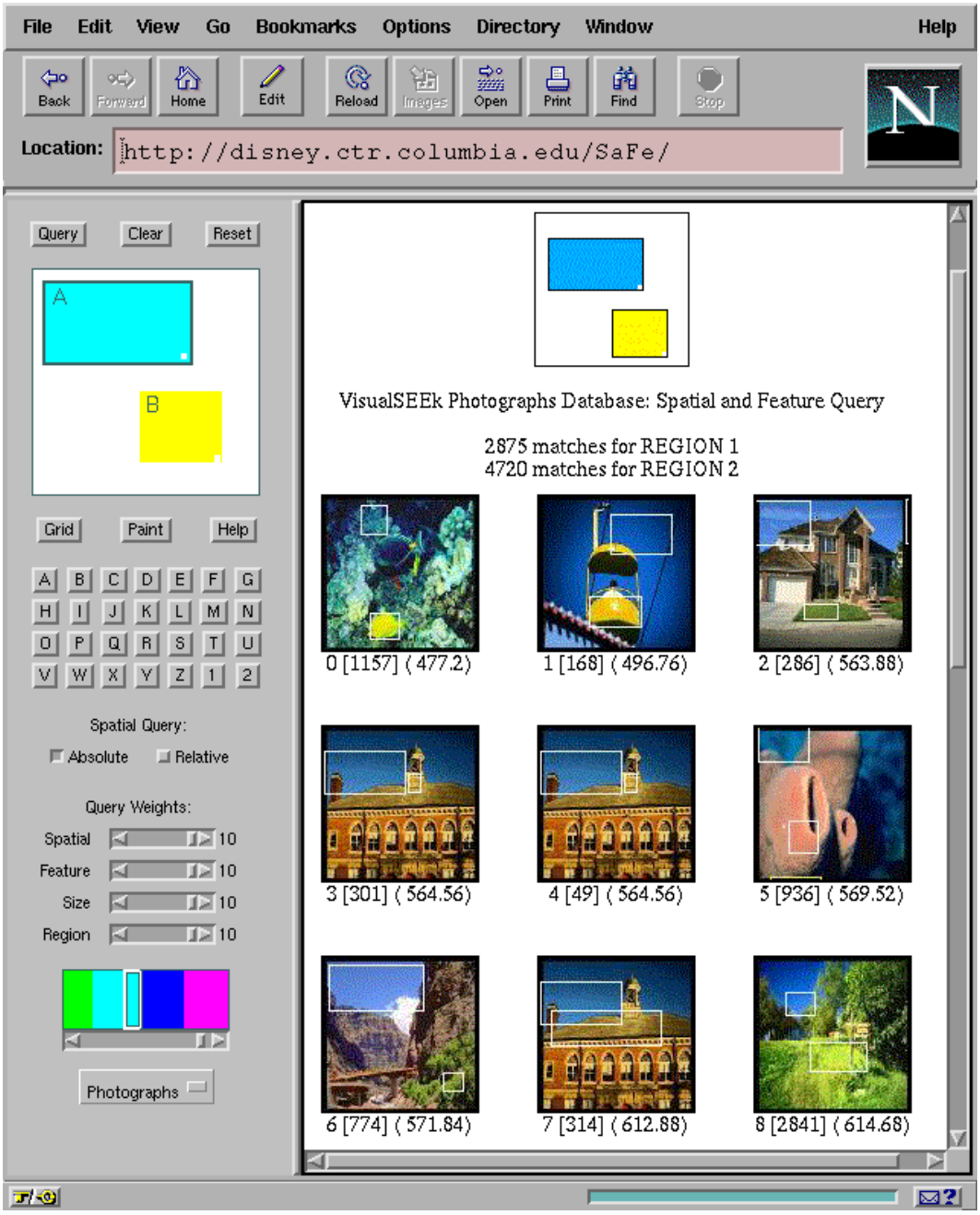


Figure 7-21: SaFe user interface to the photographic image query system.

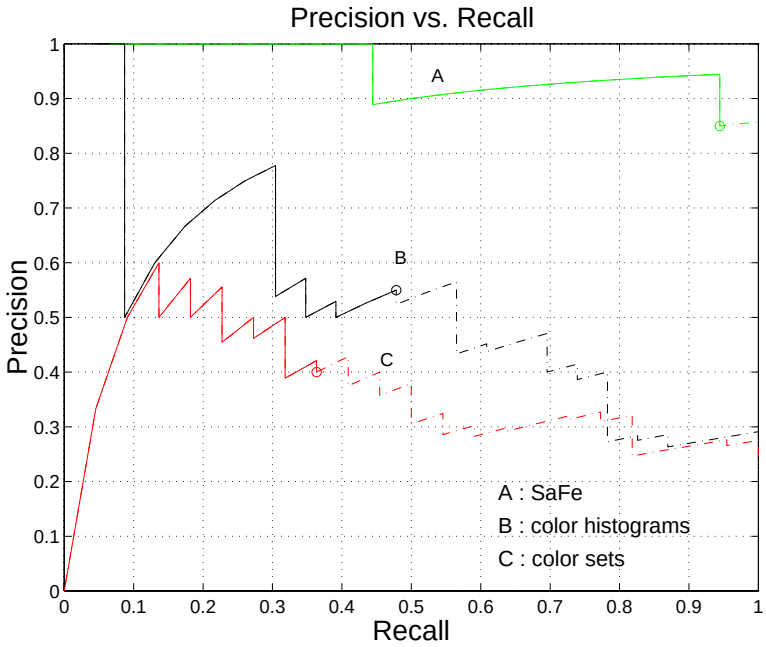


Figure 7-22: Query 1. Sunset image retrievals ($\mathcal{A}$ = SaFe query, $\mathcal{B}$ = color histograms, $\mathcal{C}$ = color sets).

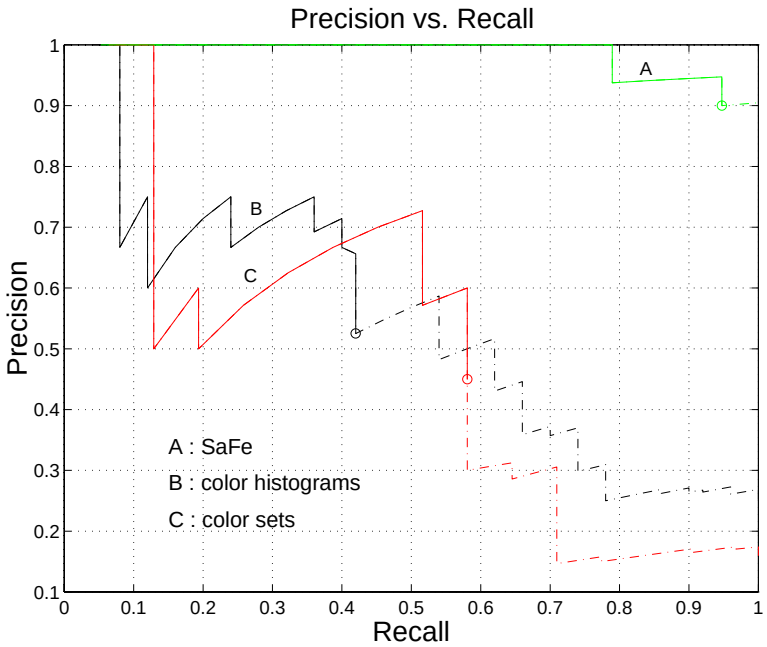


Figure 7-23: Query 2. Blue sky with greenery image retrievals ($\mathcal{A}$ = SaFe query, $\mathcal{B}$ = color histograms, $\mathcal{C}$ = color sets).

7.4.6.2 Query 2: Blue sky images with greenery

The goal of this query is to retrieve images depicting blue sky images with greenery. Prior to the trials, the 3,100 images were inspected and each was assigned a relevance. The query results are shown in Figure 7-23.

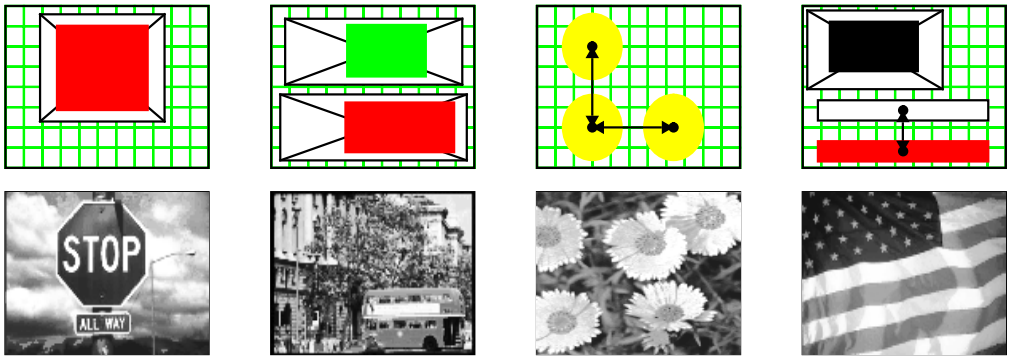
The SaFe sunset query (denoted by $\mathcal{A}$) specifies two regions (upper is blue and lower is dark green) and their spatial layout (top left of Figures 7-23). The best matches (illustrated on top from left to right) have a similar arrangement of similarly colored regions.

For the color histogram (denoted by $\mathcal{B}$) and color set (denoted by $\mathcal{C}$) queries, the best match sunset image from $\mathcal{A}$ is used as the seed image. Query $\mathcal{B}$ is computed using the histogram quadratic distance metric (Eq 7.6). Query $\mathcal{C}$ is computed using the binary set quadratic distance metric (Eq 7.7). The best matches are illustrated from left to right. The plot in Figure 7-23 shows that the retrieval effectiveness for the SaFe query $\mathcal{A}$ is much higher than in queries by methods $\mathcal{B}$ and $\mathcal{C}$.

7.4.6.3 Pedagogical image queries

We now illustrate the range of possible spatial and color queries. In the first example, in Figure 7-24(a), the query (top) specifies the absolute location of a single region. The retrieved image (bottom) has the best match in color and size to the query region and falls within the "zero-distance" bound diagrammed in the query. In Figure 7-24(b), the query specifies two regions. The retrieved image has two color match regions located at the positions in the query image.

In Figure 7-24(c), the query specifies the spatial relationships of three regions. The retrieved image has three regions that best match the colors of the query regions and their spatial relationship satisfies that specified in the query. In Figure 7-24(d), the query specifies both absolute and relative locations of regions. In this query, the match to the region positioned by absolute location (top left region in query image) considers both the color and location of this region. The match to the other regions (bottom two regions in query image) at first considers only the colors of these regions. In the last stage of the query, the spatial relationships of the regions are evaluated to determine the match.



(a) (b) (c) (d)

Figure 7-24: Example queries (a) region with absolute location, (b) two regions with absolute locations, (c) multiple regions with relative locations, (d) multiple regions with both absolute and relative locations.

7.5. Summary

We presented a new paradigm for image searching which integrates spatial (region absolute and relative locations, and size) and feature querying (visual features, i.e., color). Previous content-based query systems have not provided both types of querying [102, 112, 105, 3]. Since the discrimination of images is only partly provided by global features, the system instead utilizes image regions and their features, sizes, spatial locations, and relationships in order to compare images.

In this chapter, we described the process by which image queries are decomposed into independent region queries. These results are joined to determined candidate target images. The spatial relationships in the

candidate target images are projected into 2-D string representations. These provide for efficient validation of the spatial relationships in the target images compared to the query images.

We demonstrated that by integrating content-based and spatial querying a highly functional query system is generated which allows for a wide variety of complex spatial and feature queries. We evaluated several example spatial and color queries on a test-set of synthetic images and a test-set of color photographic images. We demonstrated that retrieval effectiveness improves significantly over purely content-based methods.

In the next chapter, we describe some of the details involving the implementation of the SaFe integrated spatial and feature image query system. We also describe a related system: VisualSEEk. The VisualSEEK system demonstrates the color/spatial image query paradigm in addition to other image search methods, such as, relevance feedback searching and text-based searching. We also present the WebSEEk image and video search engine. WebSEEk demonstrates the methods of content-based and integrated spatial and feature-based searching and cataloging of images and videos from the World-Wide Web.

Chapter 8

Content-based visual query applications

8.1. Introduction

In this chapter we introduce three new prototype applications that demonstrate the powerful techniques for content-based retrieval developed in this thesis. We first present the WebSEEk system, which is a content-based image and video search engine [148, 147]. WebSEEk consists of three components: (1) a collection system for cataloging images and videos, (2) an image and video analysis system which extracts and indexes visual features of the images and videos, and (3) the query system which consists of the query engine, the search tools, and query results display tools. We demonstrate the performance of the system in cataloging, indexing and search and retrieval of over one-half million images and videos collected from the World-Wide Web.

We also present the VisualSEEk color and spatial image query system [151, 146]. VisualSEEk provides several methods for searching for images including, integrated color and spatial querying, relevance feedback querying and text-based searching. VisualSEEk also provides a system for clustering the images to facilitate the annotation process for large collections of images.

Finally, we present the SaFe spatial and feature query system [142]. SaFe is an open-platform for computing powerful image queries that specify spatial arrangements of objects or regions and region feature properties. SaFe allows the user to construct a query image that consists of regions constrained by absolute and/or relative spatial locations. Potential applications of the SaFe system include geographic image systems (GIS), map retrieval [154], satellite image systems [87, 92] and photographic image databases [3, 45, 145].

8.2. WebSEEk image and video search engine

We first describe a system prototype for searching for images and videos on the World-Wide Web. New visual information in the form of images, graphics, animations and videos is being published on the Web at an incredible rate. However, cataloging it is beyond the capabilities of current text-based Web search engines. We describe a complete system by which visual information on the Web is

1. collected by automated agents,
2. processed in both text and visual feature domains,
3. catalogued and
4. indexed for fast search and retrieval.

We introduce an image and video search engine which utilizes both text-based navigation and content-based technology for searching visually through the catalogued images and videos. Finally, we provide an initial evaluation based upon the cataloging of over one half million images and videos collected from the Web.

8.2.1 Overview of search engines

A large number of search engines index the plethora of documents on the World-Wide Web. For example, recent systems such as Lycos, Alta Vista, Infoseek and Excite index Web documents by their textual content. These systems periodically scan the Web, analyze the documents, and create compact and searchable indexes.

The user, by entering query terms and/or by selecting subjects, uses these search engines to more easily find the desired Web documents.

Visual information is published on the Web in the form of images, graphics, bitmaps, animations and videos. As with Web documents in general, the publication of visual information is highly volatile. New images and videos are added everyday and others are replaced or removed entirely. In order to catalog the visual information, a highly efficient automated system is needed that regularly traverses the Web, detects visual information, processes it and indexes it in such a way to allow for efficient and effective search and retrieval.

We have developed a prototype image and video search engine[†] to fulfill this need [148, 147]. The system collects images and videos from the Web, catalogs them, and provides tools for searching and browsing. The system is novel in that it utilizes text and visual information synergically to provide for cataloging and searching. The complete system possesses several powerful functionalities, namely,

1. automated collection of visual information from the Web,
2. automated image and video subject classification,
3. searching using innovative visual content-based techniques,
4. image and video subject search and navigation, and text-based searching,
5. compact presentation of images and videos for displaying query results,
6. search results lists manipulations such as intersection, subtraction and concatenation, and
7. query modification using content-based relevance feedback.

Recently, several World-Wide Web image search engines have been reported [134, 148, 47, 70]. The Webseer system from University of Chicago [47] provides a system for searching for images and animations. The system is novel in that it detects faces within the images and lets the users search by the number of faces. The Webseer system does not classify the images into subject categories and does not provide for content-based searching. The Interpix [70] image search engine provides for content-based image searching using color histograms. The Interpix system has recently been integrated with Yahoo's search engine allowing for retrieval of images related to some of Yahoo's Web categories.

8.2.2 Image and video collection process

The image and video collection process is conducted by autonomous Web agents or "spiders." The spiders traverse the Web by following the hyperlinks between documents. They detect images and videos, retrieve and process them and add the new information to the catalog. The overall collection process, illustrated in Figure 8-1, is carried out using several distinct modules:

1. the *Traversal Spider* – assembles lists of candidate Web documents that include images, videos or hyperlinks to them,
2. the *Hyperlink Parser* – extracts the Web addresses (*URL*s) of the images and videos, and
3. the *Content Spider* – retrieves, analyzes and iconifies the images and videos.

8.2.2.1 Image and video detection

In the first phase, the *Traversal Spider* traverses the Web looking for images and videos, as illustrated in Figure 8-2. Starting from seed *URL*s, the *Traversal Spider* follows a breadth-first search path. It retrieves Web documents via Hypertext Transfer Protocol (*HTTP*) and passes the Hypertext Markup Language (*HTML*) code to the *Hyperlink Parser*. In turn, the *Hyperlink Parser* detects new *URL*s, encoded as *HTML* hyperlinks, and adds them back to the queue of Web documents to be retrieved by the *Traversal Spider*.

[†] `http://www.ctr.columbia.edu/webseek`

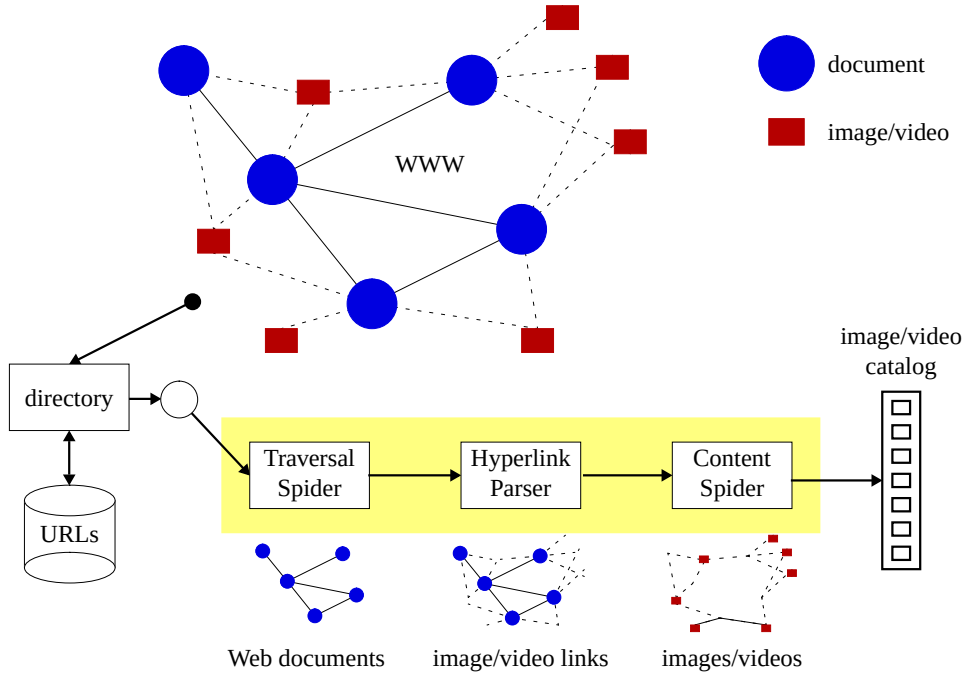


Figure 8-1: Image and video gathering process.

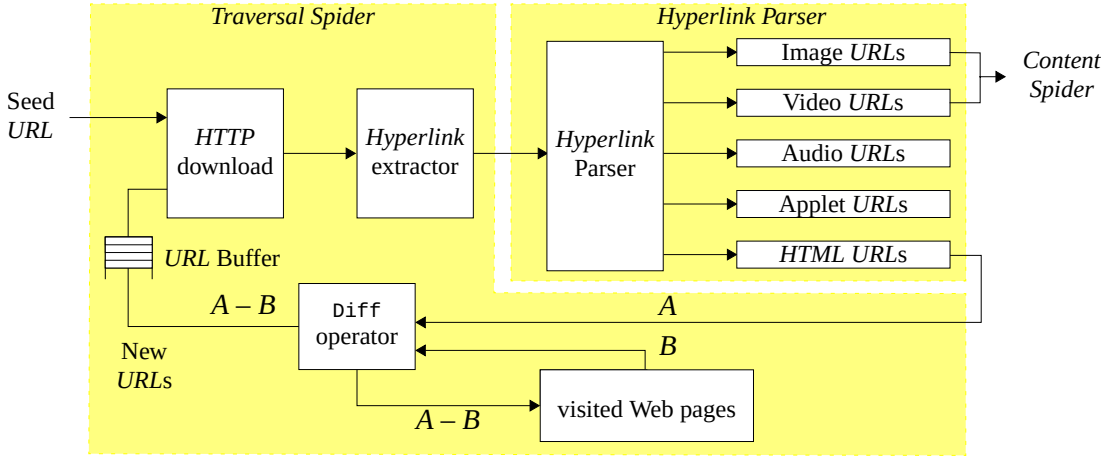


Figure 8-2: The *Traversal Spider* traverses the Web and assembles the list of *URL*s of images and videos.

The *Hyperlink Parser* detects the hyperlinks in the Web documents and converts the relative *URL*s into absolute addresses. By examining the types of the hyperlinks and the filename extensions of the *URL*s, the *Hyperlink Parser* assigns each *URL* to one of several categories: image, video or *HTML*. The mapping between filename extensions and Web object type is given by the *Multipurpose Internet Mail Extensions* (MIME) content type labels, as illustrated in Table 8.1.

Extension	Type	Description
.gif	image	Compuserve image format
.jpg, .jpeg, .jpe, .jfif, .pjpeg, .pjp	image	JPEG image format
.qt, .mov, .moov	video	Quicktime video format
.mpeg, .mpg, .mpr, .mpv, .vbs, .mpegv	video	MPEG video format
.avi	video	Microsoft video format
.htm, .html	*HTML*	Hypertext markup language

Table 8.1: MIME mapping between extensions and object types.

In the second phase, the list of image and video *URL*s from the *Hyperlink Parser* is passed to the *Content Spider*. The *Content Spider* retrieves the images and videos, processes them and adds them to the WebSEEk catalog. Three important functions of the *Content Spider* are to

1. extract visual features that allow for content-based techniques in searching, browsing and grouping,
2. extract other attributes such as width, height, number of frames, type of visual data (i.e., photo, graphic, video, etc.), and so forth, and
3. generate an icon, or motion icon, that sufficiently compacts and represents the visual information to be used for browsing and displaying query results.

The process of extracting visual features generates a color histogram and a table of extracted color regions for each image and video. The process is discussed in more detail in Section 8.2.5. The other attributes of the images and videos populate the WebSEEk database tables, which are defined in Section 8.2.3.4. Finally, the *Content Spider* generates coarse and highly compressed versions of the images and videos to provide pictorial data in the query output.

Image and video presentation The image icons are obtained by spatially reducing the original images. The video icons are generated by reducing the original videos both spatially and temporally. The temporal reduction is achieved in a two step process: first, only one frame is retained per every one second of video. Next, the key frames of the sequence are identified using automated shot detection [95]. Finally, the video is re-animated from the key frames and packaged as an animated-GIF file.

8.2.3 Subject classification process

The utilization of text is essential to the cataloging process. In particular, every image and video on the Web has a unique Web address (*URL*) and possibly other *HTML* tags. This text potentially provides a semantic description of the corresponding visual information. We process the *URL*s and *HTML* tags in the following ways to index the images and videos:

- extract **terms** t_k's,
- extract **directory names** d_l's,
- map **key-terms** t_k^*'s to **subjects** s_m's using a **key-term dictionary** $\mathcal{M}_{km}$, and
- map **directory names** d_l's to **subjects** s_m's.

8.2.3.1 Text processing

The images and videos are published on the Web by two distinct methods: *inline* and *reference*. The *HTML* syntax differs in the two cases:

1. To inline, or embed, an image or video into a Web document, the following code is included in the document: `<img src=URL alt=[alt text]>`, where `URL` gives the relative or absolute address of the image or video. The optional `alt` tag specifies the text that substitutes for the inlined image or video when the image or video is not displayed.

2. Alternatively, images and videos are referenced from parent Web documents using the following code: `<a href=URL>[hyperlink text]</a>`, where the optional `[hyperlink text]` provides the high-lighted text that describes the image/video pointed to by the hyperlink.

Term extraction The **terms**, t_k's, are extracted from the image and video *URL*s, *alt* tags and hyperlink text by chopping the text at non-alpha characters. For example, the *URL* of an image or video has the following form

`URL = http://host.site.domain[:port]/[user/][directory/][file[.extension]].`

Here `[...]` denotes an optional argument. For example, several typical *URL*s are

$\text{URL}_1 =$ `http://www.mynet.net:80/animals/domestic-beasts/dog37.jpg`,
$\text{URL}_2 =$ `http://camille.gsfc.nasa.gov/rsd/movies2/Shuttle.gif`,
$\text{URL}_3 =$ `http://www.arch.columbia.edu/DDL/projects/amiens/slide6b.gif`.

Terms are extracted from the `directory` and `file` strings using functions $\mathcal{F}_{\text{key}}$ and $\mathcal{F}_{\text{chop}}$ where

$$\mathcal{F}_{\text{key}}(\text{URL}) = \mathcal{F}_{\text{chop}}(\texttt{directory/file}),$$

and $\mathcal{F}_{\text{chop}}(\text{string})$ gives the set of substrings that are delimited by non-alpha characters. For example,

$$\begin{aligned} \mathcal{F}_{\text{key}}(\text{URL}_1) &= \mathcal{F}_{\text{chop}}(\text{“animals/domestic-beasts1/dog37”}) \\ &= \text{“animals,” “domestic,” “beasts,” “dog”.} \end{aligned}$$

The process of extracting terms produces an overall set of terms $\{t_k\}$ for the image and video collection. The system indexes the images and videos directly using $\{t_k\}$. In addition, certain terms, key-terms, t_k^*'s, are used to map the images and videos to subject classes, as we explain shortly.

Directory name extraction A **directory name**, d_l, is a phrase extracted from the *URL*s that groups images and videos by location on the Web. The directory name consists of the directory portion of the *URL*, namely, $\mathcal{F}_{\text{dir}}(\text{URL}) =$ `directory`. For example, $\mathcal{F}_{\text{dir}}(\text{URL}_1) =$ “animals/domestic-beasts”. The process of extracting directory names produces an overall set of directory names $\{d_l\}$ for the image and video collection. The directory names are also used by the system to map images and videos to subject classes.

8.2.3.2 Image and video subject taxonomy

A **subject class** or **subject**, s_m, is an ontological concept that represents the semantic content of an image or video, i.e., subject = “The Beatles.” A **subject taxonomy** is an arrangement of the set of subject classes $\{s_m\}$ into a hierarchy, denoted as $\widehat{\{s_m\}}$, i.e., “The Beatles” $\in$ “Rock music” $\in$ “Music.” We are developing the subject taxonomy for image and video topics; a portion is illustrated in Figure 8-3. When we detect potentially descriptive terms, such as $t_k =$“dog” or $t_k =$“canine,” we add an appropriate subject class, i.e., $s_m =$“animals/dogs,” to the subject taxonomy.

8.2.3.3 Key-term dictionary

The **key-terms**, t_k^*'s, are terms that are manually identified to be semantically related to one or more subject classes, s_m's. The **key-term dictionary** contains the set of key-terms $\{t_k^*\}$ and the related subject classes $\{s_m\}$. As such, the key-term dictionary provides a set of mappings $\{\mathcal{M}_{km}\}$ from key-terms to subject classes, where $\mathcal{M}_{km} : t_k^* \rightarrow s_m$.

We build the key-term dictionary in a semi-automated process. In the first stage, the term histogram for the image and video archive is computed such that each term t_k is assigned a count number f_k which indicates how many times the term appeared. Then, the terms are ranked by highest count f_k and are presented in this order of priority for manual assessment.

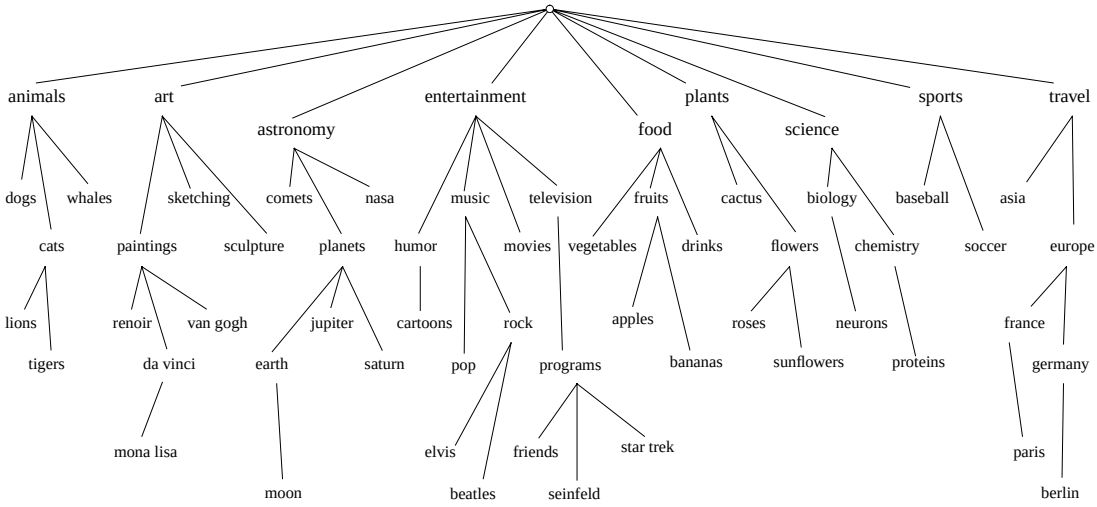


Figure 8-3: Portion of the image and video subject taxonomy $\{\widehat{s_m}\}$.

The goal of the manual assessment is to assign qualified terms $t_k^* \in \{t_k\}$ to the key-term dictionary. The qualified terms should be descriptive and not homonymic. Ambiguous terms make poor key-terms. For example, the term "rock" is a not a good key-term due to its ambiguity. "Rock" refers to either a large mass of stone, rock music, a diamond, or several other things. Once a term and its mappings are added to the key-term dictionary, it is used to assign the images and videos in the collection to subject classes.

term: t_k	count: f_k
image	86380
gif	28580
icon	14798
pic	14035
img	14011
graphic	10320
picture	10026
small	9442
art	8577

(a)

key-term: t_k^*	count: f_k	mapping $\mathcal{M}_{km}$ to subject s_m
planet	1175	astronomy/planets
music	922	entertainment/music
aircraft	458	transportation/aircraft
travel	344	travel
gorilla	273	animals/gorillas
starwars	204	entertainment/movies/films/starwars
soccer	195	sports/soccer
dinosaur	180	animals/dinosaurs
porsche	139	transportation/automobiles/porsches

(b)

Table 8.2: Sample (a) terms and their counts $\{t_k : f_k\}$ and (b) key-terms counts $\{t_k^* : f_k\}$ with subjects s_m's and mappings $\mathcal{M}_{km} : t_k^* \rightarrow s_m$'s. Taken from the assessment of over 500,000 images and videos.

1 From the initial experiments of cataloging over 500,000 images and videos, the terms listed in Table 8.2 are a sample of those extracted. Notice in Table 8.2(a) that some of the most frequent terms are not sufficiently descriptive of the visual information, i.e., terms: "image," "picture." The terms in Table 8.2(b) unambiguously define the subject of the images and videos, i.e., terms: "aircraft," "gorilla," "porsche."

In a similar process, the directory names d_l's are mapped manually to subject classes s_m's. In this case, an entire directory of images/videos may be mapped to one or more subject classes. For example, the directory d_l = "space/planets/saturn" is mapped to subject s_m = "astronomy/planets/saturn." Similar to the process for key-term extraction, the system computes a histogram of directory names $\{d_l : f_l\}$ and presents it for manual inspection. The directories that sufficiently group images and videos related to particular topics are then mapped to the appropriate subject classes.

In Section 8.2.6.1, we demonstrate that these methods of key-term and directory name extraction coupled with subject mapping provide excellent performance in classifying the images and videos by subject. By incorporating some results of natural language processing in addition to using visual features, we hope to further improve and automate the subject classification process.

8.2.3.4 Catalog database

As described above, each retrieved image and video is processed and the following information tables are populated:

IMAGES – IMID URL NAME FORMAT WIDTH HEIGHT FRAMES DATE
TYPES – IMID TYPE
SUBJECTS – IMID SUBJECT
TEXT – IMID TERM
FV – IMID COLOR-HISTOGRAM
REGIONS – IMID REGID COLOR-SET X Y WIDTH HEIGHT

where the special (non-alphanumeric) data types are given as follows:

TYPE $\in$ {Color photo, Color graphic, Video, B/w image, Gray image}
SUBJECT $\in$ {Subject classes from taxonomy $\{\widehat{s_m}\}$, partially depicted in Figure 8-3}
COLOR-HISTOGRAM $\in$ $\mathcal{R}^{166}$ (166-bin histogram)
COLOR-SET $\in$ $\mathcal{B}^{166}$ (166-element binary set).

The automated assignment of TYPE to the images and videos using visual features is explained in Section 8.2.5.3. Queries on the database tables: IMAGES, TYPES, SUBJECTS and TEXT are performed using standard relational algebra. For example, the query: Give me all records with TYPE = "video", SUBJECT = "news" and TERM = "basketball" can be carried in SQL as follows:

SELECT IMID
FROM TYPES, SUBJECTS, TEXT
WHERE TYPE = "video" **AND** SUBJECT = "news" **AND** TERM = "basketball."

However, content-based queries, which involve table FV and REGIONS, require special processing, which is discussed in Sections 8.2.4.2 and 8.2.5.

8.2.4 Search and retrieval processes

To retrieve images and videos the user may enter search terms t_k's or select an image/video subject s_m. The search process and model for user-interaction is depicted in Figure 8-4. The query for images and videos produces search results **A**. For example, Figure 8-5 illustrates the search results for a query for images and videos related to s_m ="nature," that is, **A** = Query(SUBJECT = "nature"). The user may manipulate, search or view **A**.

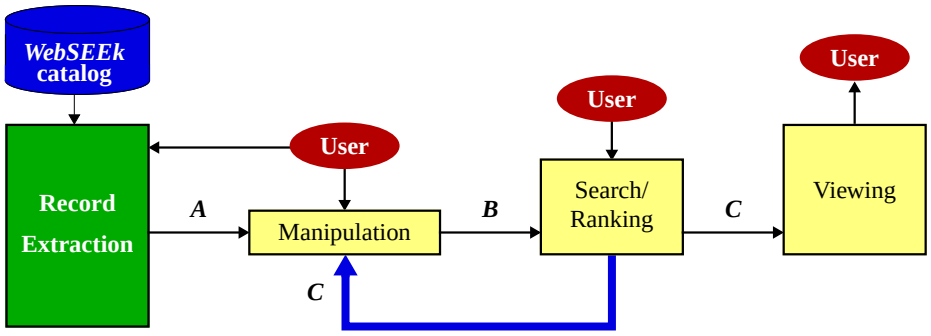


Figure 8-4: Search, retrieval and search results list manipulation processes.

8.2.4.1 Search results list manipulation

The results of a prior search may be fed-back to the manipulation module as list **C**, as illustrated in Figure 8-4. The user manipulates **C** by adding or removing records. This is done by issuing a new query that creates a second, intermediate list **A**. The user then generates the new list **B** by selecting one of the

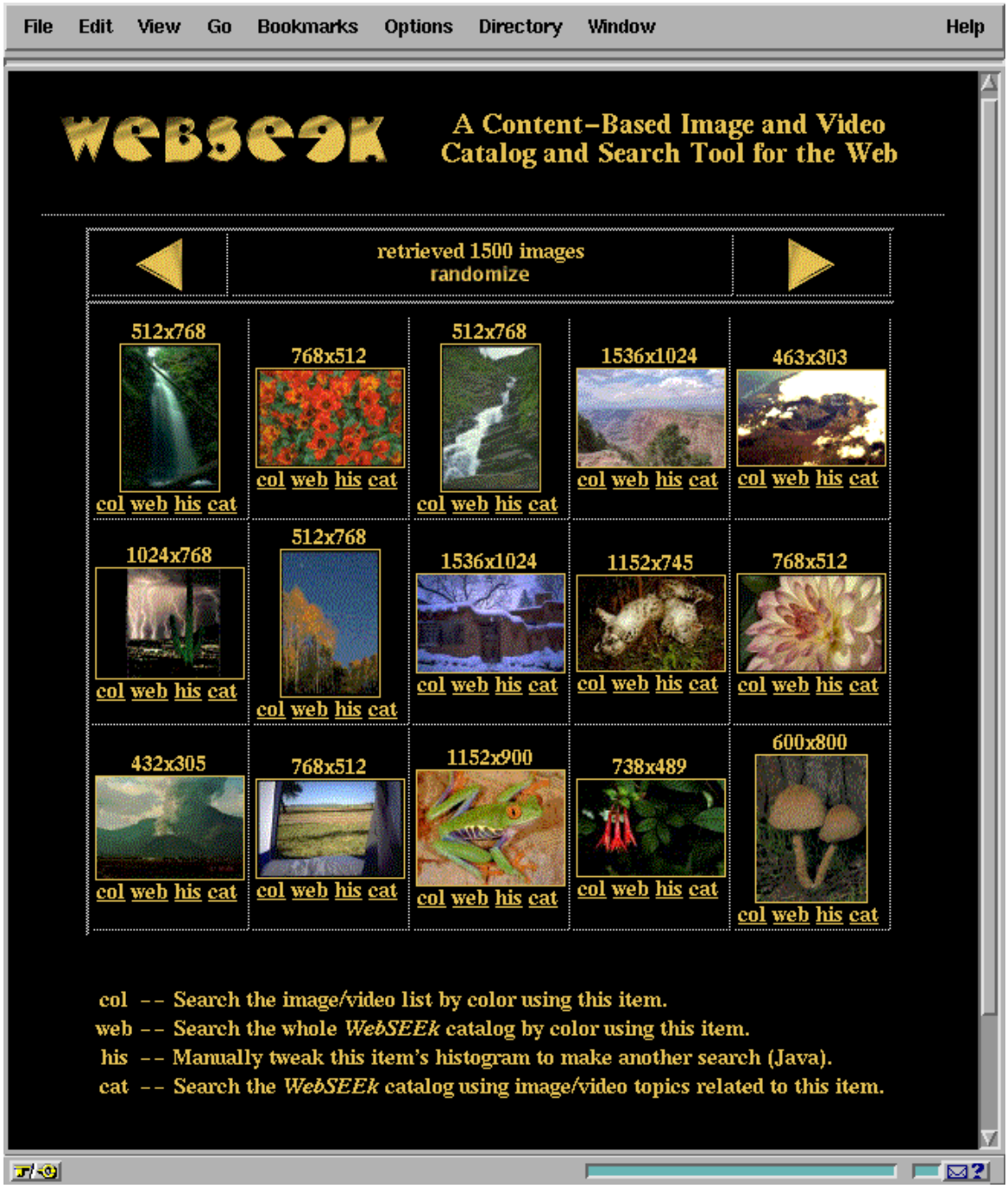


Figure 8-5: Search results for `SUBJECT` = "nature."

following manipulations on **C** using **A**, for example, define **C** = Query(`SUBJECT` = "nature") and **A** = Query(`TERM` = "sunset"), then

union:	$\mathbf{B} = \mathbf{A} \cup \mathbf{C}$,	i.e., **B** = Query(`TERM` = "sunset" or `SUBJECT` = "nature"),
intersection:	$\mathbf{B} = \mathbf{A} \cap \mathbf{C}$,	i.e., **B** = Query(`TERM` = "sunset" and `SUBJECT` = "nature"),
subtraction:	$\mathbf{B} = \mathbf{C} - \mathbf{A}$,	i.e., **B** = Query(`SUBJECT` = "nature" and `TERM` $\neq$ "sunset"),
replacement:	$\mathbf{B} = \mathbf{A}$,	i.e., **B** = Query(`TERM` = "sunset"),
keep:	$\mathbf{B} = \mathbf{C}$,	i.e., **B** = Query(`SUBJECT` = "nature").

8.2.4.2 Content-based visual query

The user may browse and search the list **B** using both content-based and text-based methods. In the case of content-based searching, the output is a new list **C**, where $\mathbf{C} \subseteq \mathbf{B}$ is an ordered subset of **B** such that **C** is ordered by highest similarity to the image/video selected by the user (as described in Section 8.2.5). The search and browse operations may be conducted on the input list **B** or the entire catalog.

For example, $\mathbf{C} = \mathbf{B} \simeq \mathbf{B}^{\text{sel}}$, where $\simeq$ means "visually similar," ranks list **B** in order of highest similarity to the selected item $\mathbf{B}^{\text{sel}}$. For example, the following content-based visual query:

$$\mathbf{C} = \text{Query}(\texttt{SUBJECT} = \text{"nature"}) \simeq \mathbf{B}^{\text{sel}}(\text{"mountain scene image"}),$$

ranks the "nature" images and videos in order of highest visual similarity to the selected "mountain scene image." In the example illustrated in Figure 8-6, the query $\mathbf{C} = \mathbf{B} \simeq \mathbf{B}^{\text{sel}}$("red race car"), retrieves the images and videos from the subject class: `SUBJECT` = "transportation/automobiles/ferraris" that are visually similar to the selected image of a "red race car."

Alternatively, the user can select one of the items in the current search results list **B** and then use it to search the entire catalog for similar items. For example, $\mathbf{C} = \mathbf{A} \simeq \mathbf{B}^{\text{sel}}$ ranks list **A**, where in this case **A** is the full catalog, in order of highest visual similarity to the selected item from **B**.

8.2.5 Content-based techniques

The system provides two methods for content-based searching: (1) by color histograms, and (2) by spatial locations and arrangements of color regions. We adopted these techniques in order to utilize domain-independent approaches [146]. The content-based methods developed here for indexing, searching and navigation can be applied, in principle, to other types of features, such as texture, motion and so forth, and to other application domains.

8.2.5.1 Global feature query

The color histograms describe the distribution of colors in each image or video. We define the color histograms as discrete, 166-bin, distributions in a quantized *HSV* color space [146]. The system computes a color histogram for each image and video scene, which is used to assess its similarity to other images and video scenes. The color histograms are also used to automatically assign the images and videos to type classes using Fisher discriminant analysis, as described in Section 8.2.5.3.

Color histograms dissimilarity Color histograms are used to measure the dissimilarity of the color contents of the images. A histogram dissimilarity metric determines the dissimilarity between a query histogram and a target histogram [45, 3]. In order to achieve high efficiency in the color histogram query process, we utilize the technique of query optimized distance (QOD) computation. This technique provides both efficient indexing, retrieval of target histograms and computation of the histogram dissimilarities. In the QOD technique, the retrieval of candidate target images is prioritized in the order of the most significant colors in the query histogram.

We have found that setting the "color significance" threshold adaptively to select only the $80 - 90\%$ most significant colors in the query histogram decreases query time dramatically without degrading retrieval effectiveness. By using the efficient QOD computation method, we are able to greatly reduce the query processing time, as demonstrated in Section 8.2.6.3.

Figure 8-6: Content-based visual query results for images/videos $\simeq$ "red race car".

8.2.5.2 Integrated spatial and color query

While color histograms are useful for indexing images by global color, a significant aspect of discriminating among images depends on the sizes, spatial locations and relationships of objects or regions within the images. Therefore, we developed a system for querying by spatial locations and arrangements of color regions [146]. With this integrated spatial and feature image query engine, the user graphically constructs a query by placing color regions on a query grid, as illustrated in Figure 8-7.

The similarity between the query image and each target image is computed by assessing the similarities of the spatial locations, colors and sizes of the regions in the query and target images. The integrated spatial and features system provides greater power and flexibility in querying for images than the color histogram-based methods [146]. We found that retrieval effectiveness improves substantially using this method.

8.2.5.3 Automated type assessment

By training on samples of the color histograms of images and videos, we developed a process of automated type assessment using Fisher discriminant analysis (FDA). FDA constructs a series of uncorrelated linear weightings of the color histograms that provide for maximum separation between training classes [48]. New color histograms are automatically assigned to nearest Type $\in$ {color photo, color graphic, video, b/w image, gray image} class [148].

In Section 8.2.6.2, we show that this approach provides for excellent performance in automatically classifying the images and videos into these broad type classes. We hope to increase the number of type classes and improve the type-classification system by incorporating other visual features such as spatial and color region information into the classification process.

8.2.5.4 Relevance feedback

The user best determines from the results of a query which images and videos are relevant and not relevant. The system can use this information to reformulate the query to better retrieve the images and videos the user desires [77]. Using the color histograms, relevance feedback is accomplished as follows: let I_r = {relevant images/videos} and I_n = {non-relevant images/videos} as determined by the user. The new query vector $\mathbf{h}_q^{k+1}$ at round $k+1$ is generated by

$$\mathbf{h}_q^{k+1} = ||\alpha \mathbf{h}_q^k + \beta \sum_{i \in I_r} \mathbf{h}_i - \gamma \sum_{j \in I_n} \mathbf{h}_j||, \tag{8.1}$$

where $||\cdot||$ indicates normalization. The new images and videos are retrieved using $\mathbf{h}_q^{k+1}$. One formulation of relevance feedback assigns the values $\alpha = 0$, and $\beta = \gamma = 1$, which weights the positive and negative examples equally. The process of selecting the example images for content-based relevance feedback searching is illustrated in Figure 8-8(a). A simpler form of relevance feedback allows the user to select only one positive example in order to iterate the query process. In this case, $\alpha = \gamma = 0$, $\beta = 1$, $|I_r| = 1$ and $|I_{nr}| = 0$ gives the new query vector directly from the selected image/video's color histogram, $\mathbf{h}_{I_r}$ as follows, $\mathbf{h}_q^{k+1} = \mathbf{h}_{I_r}$.

8.2.5.5 Histogram manipulation

The system also provides a tool for the user to directly manipulate the image and video color histograms to formulate the search. Using the histogram manipulation tool, illustrated in Figure 8-8(b), the user may select one of the images or videos from the results and display its histogram. The user can then modify the histogram by adding or removing colors. The modified histogram is then used to conduct the next search. The new query histogram $\mathbf{h}_q^{k+1}$ is generated from a selected histogram $\mathbf{h}_s$ by adding or removing colors, which are denoted in the modifications histogram $\mathbf{h}_m$,

$$\mathbf{h}_q^{k+1} = ||\mathbf{h}_s + \mathbf{h}_m||. \tag{8.2}$$

8.2.6 Evaluation

In the initial trials, the system has catalogued 513,323 images and videos from 46,551 directories at 16,773 Web sites. The process required several months, and was performed simultaneously with the development of the user application. Overall the system has catalogued over 129 Gigabytes of visual information. The

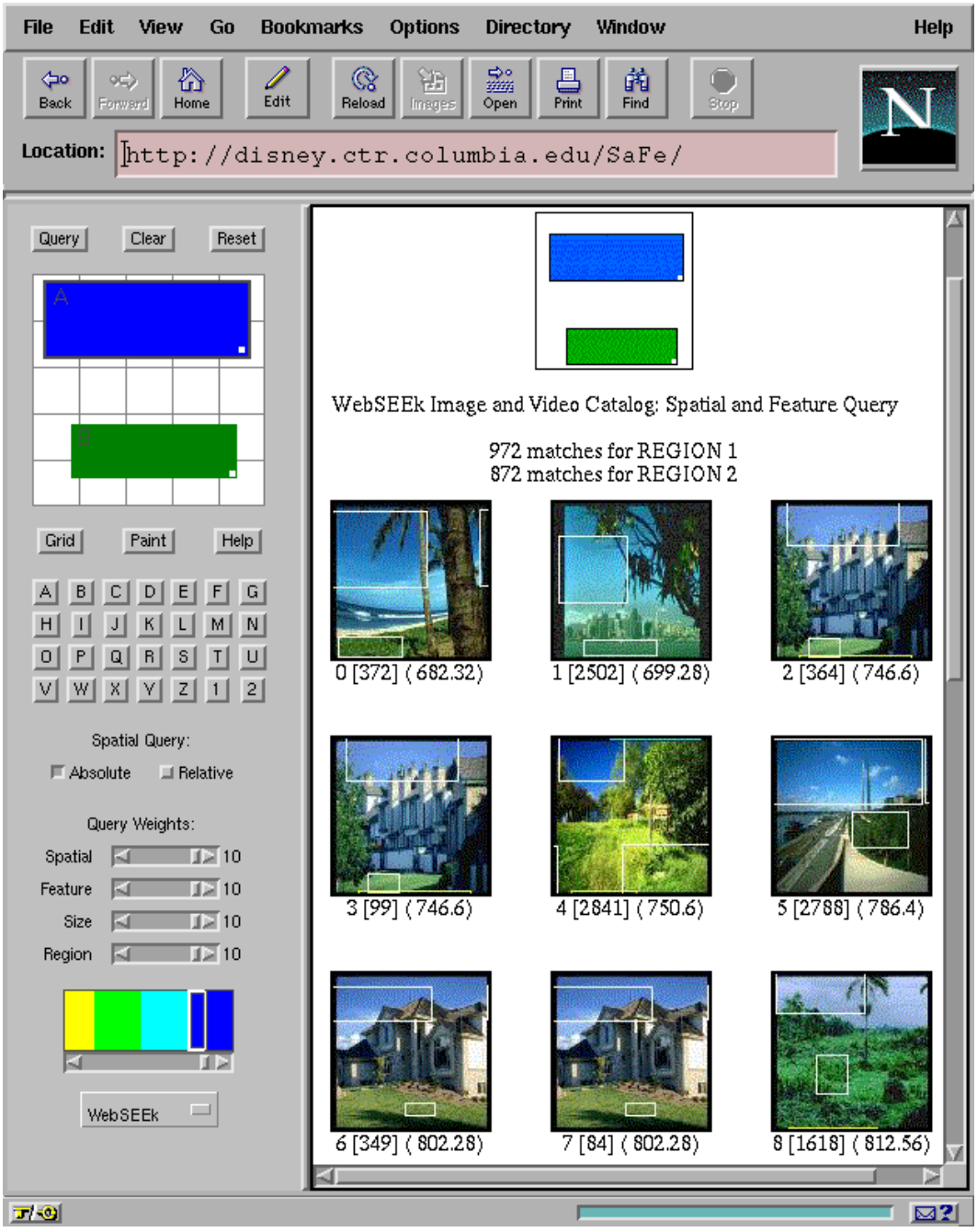


Figure 8-7: WebSEEk utilizes the SaFe integrated spatial and feature image search tools. SaFe finds the images which contain similar regions in similar spatial arrangements to the query.

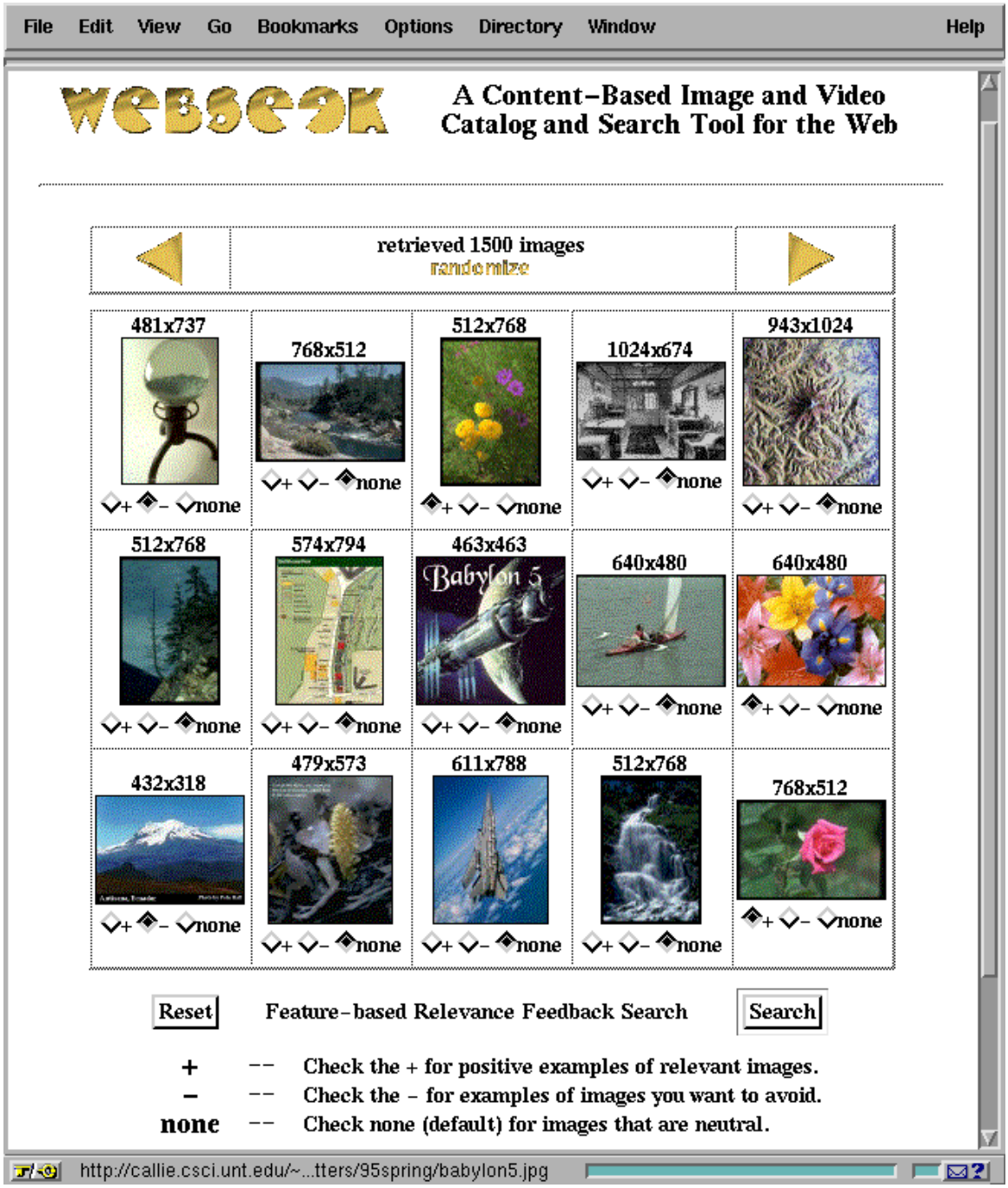


Figure 8-8: Relevance feedback search allows user to select both positive and negative examples.

local storage of information, which includes coarse versions of the data and color histogram feature vectors, requires approximately 2 Gigabytes.

8.2.6.1 Subject classification evaluation

The cataloging process assigned 68.23% of the images and videos into subject classes using key-term and directory name mappings. To evaluate the subject classification system we assessed the classification rates for several subject classes. The results are summarized in Table 8.3. The classification performance is excellent; the system provides a classification precision of approximately 92%. For this assessment we chose nine classes, depicted in Table 8.3, at random from the subject taxonomy of 2128 classes. We established the ground-truth by manually verifying the subject of each image and video in the nine test sets.

We observed that errors in classification result from several occurrences:

1. the key-terms defined by the WebSEEk system are sometimes used out of context by the publishers of the images or videos,
2. the system relies on some ambiguous key-terms, i.e., t_k^* ="madonna" and
3. the semantics of the images and videos change when viewed outside of the context of their Web sites.

For example, in this last category the precision of subject class s_m ="animals/possums" is low, as depicted in Table 8.3, because five out of the nine items are not images or videos of possums. These items were classified incorrectly because the key-term "possum" appeared in the directory name. While some of the images in that directory depict possums, others depict only the forests to which the possum are indigenous. When viewed outside of the context of the "possum" Web site, the images of forests are not expected to be in the class "animals/possums."

Subject	# Sites	Count	Rate
art/illustrations	29	1410	0.978
entertainment/humour/cartoons/daffyduck	14	23	1.000
animals/possums	2	9	0.444
science/chemistry/proteins	7	40	1.000
nature/weather/snow/frosty	9	13	1.000
food	403	2460	0.833
art/paintings/pissarro	3	54	1.000
entertainment/music/mtv	15	87	0.989
horror	366	2454	0.968

Table 8.3: Rates of correct subject classification (precision) for random set of classes.

8.2.6.2 Type classification evaluation

The precision of the automated type classification system is summarized in Table 8.4. For this evaluation, the training and test samples consist of 200 images from each type class. We found the automated type assessment for these five simple classes is excellent. The system provides a classification precision of 95%. In future work, we will extend WebSEEk to include a larger number of classes, including new type classes, such as *Fractal* images, *Cartoons*, *Faces*, *Art paintings* and subject classes from the subject taxonomy.

Type	Rate
Color photo	0.914
Color graphic	0.923
Gray image	0.967
B/w image	1.000

Table 8.4: Rates of correct automated type classification.

8.2.6.3 Efficiency

Another important factor in the image and video search system is the speed at which user operations and queries are performed. In the initial system, the overall efficiency of various database manipulation operations is excellent, even on the large catalog ($> 500,000$ images/videos). In particular, the good performance of the content-based visual query methods is provided by the strategy for indexing the 166-bin color histograms described in Section 8.2.5.1. For example, the system identifies the $\mathcal{N} = 60$ most similar visual scenes in the catalog of $513,323$ images and videos to a selected query scene in only 1.83 seconds.

We nest describe the VisualSEEk prototype image database system. VisualSEEk demonstrates the color/spatial search method in the retrieval of color photographic images. VisualSEEK also has components for text-based searching, "power" annotation of images and content-based relevance feedback searching.

8.3. VisualSEEk color/spatial image query system

VisualSEEk is an image database system that provides tools for searching, browsing and retrieving images. VisualSEEk is novel in that the user queries for images by specifying color regions and their spatial locations. We describe the VisualSEEk color/spatial image query processes, the relevance feedback color search system and the text-based search and annotation methods. We demonstrate that the relevance feedback retrieval and annotation components have critical roles in the VisualSEEk system. For example, by clustering the images by color, the annotation process is greatly accelerated. Finally, we demonstrate the query methods and show some example queries.

The goal of the VisualSEEk project is to implement a CBVQ system that provides efficient and effective retrieval of images [151]. The VisualSEEk project emphasizes several objectives in order to improve image retrieval functionality:

1. automated extraction of localized regions and colors from images [145],
2. querying by both color and spatial information [146],
3. fast indexing and retrieval, and
4. the design of highly functional user tools.

We developed the VisualSEEk client application in Java to allow for maximum functionality, client platform independence and accessibility on the World-Wide Web.

8.3.1 System design

The VisualSEEk system consists of several components, as illustrated in Figure 8-9: (1) the user tools, (2) the query server, (3) the image retrieval server, (4) the image archive, (5) the meta-data database and (6) the index files.

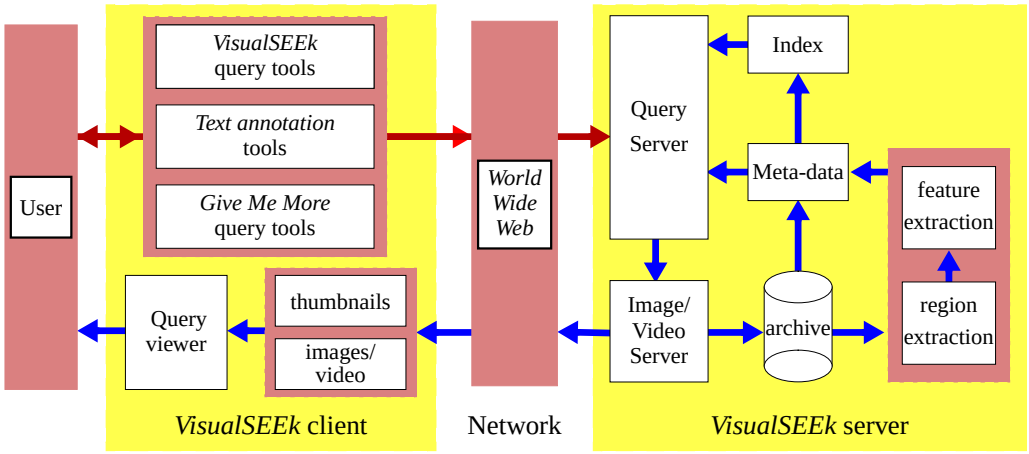


Figure 8-9: VisualSEEk client-server architecture.

These components construct the three distinct modules of the VisualSEEk system: (1) the client application, (2) the network and communication application and (3) the server application. The client application consists of a suite of Java applets that execute within the World-Wide Web browser. The VisualSEEk applet collects the query from the user and sends the query string to the server via the common gateway interface (CGI) using the hypertext transfer protocol (HTTP) as depicted in Figure 8-10. The HTTP client-server system handles VisualSEEk's communication via the World-Wide Web. The CGI component of the HTTP executes the VisualSEEk query programs on the server machine.

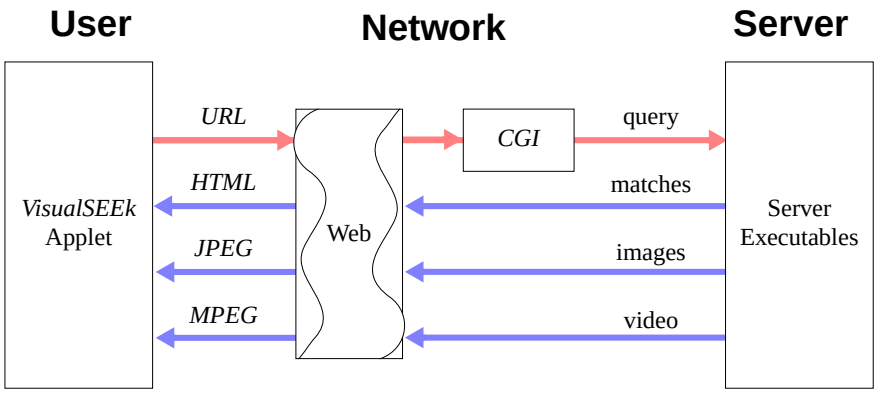


Figure 8-10: Data flow between client and server applications in the VisualSEEk system.

8.3.1.1 Client application

The client application provides query tools that allow the user to construct a query by diagramming color regions on a query grid. The user assigns the regions attributes of spatial location, size and color, as illustrated in Figure 8-11(a). The user may also assign boundaries for the locations and sizes of the regions.

Figure 8-11(a) depicts an example query that specifies three regions - blue, green and light blue, and their locations and sizes. The results of this VisualSEEk query are illustrated in Figure 8-11(b). We see that this particular query retrieves images of under-water scenes and nature with blue skies and trees.

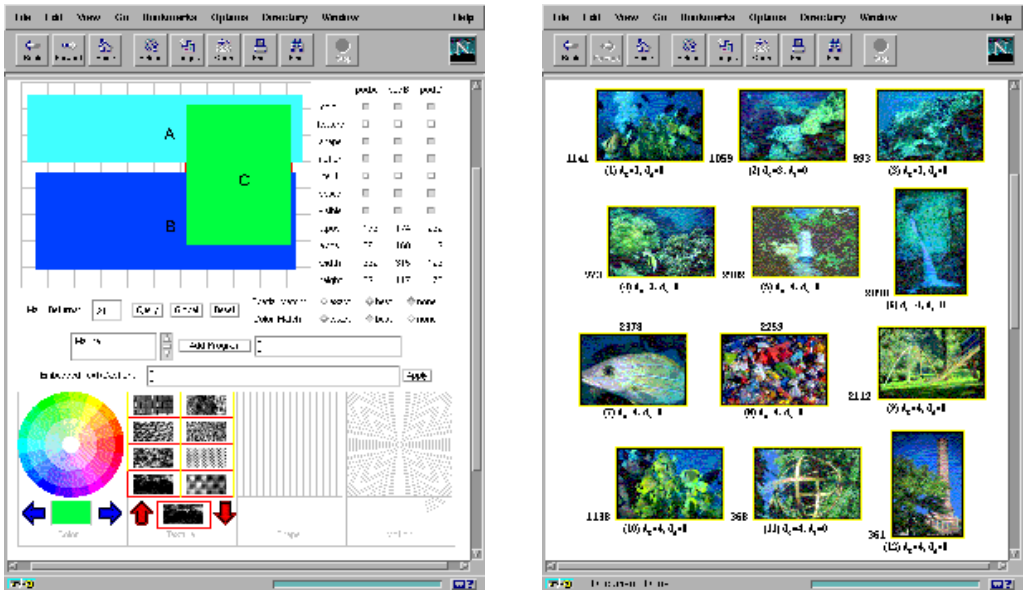
8.3.1.2 Server application

The server application computes the queries that are sent by the client application. The server application first parses the query string and decodes the number of regions and their sizes, colors and locations. Next, it generates a "mock-up" image that depicts the regions specified by the user in the query. This image is sent to the client application to be displayed as the first element in the query results page (see top of results page in Figure 8-11(b)). Then, the query server computes the query and finds the images that contain the best matches to the query regions.

The query program generates its output in the hypertext markup language (HTML) that displays the results of the query to the user. The HTML output includes thumbnail images that depict the images found in the query. The output also includes the distance scores for each retrieved image to the query image. When the user selects a thumbnail image, the system retrieves the full-size image from the image server.

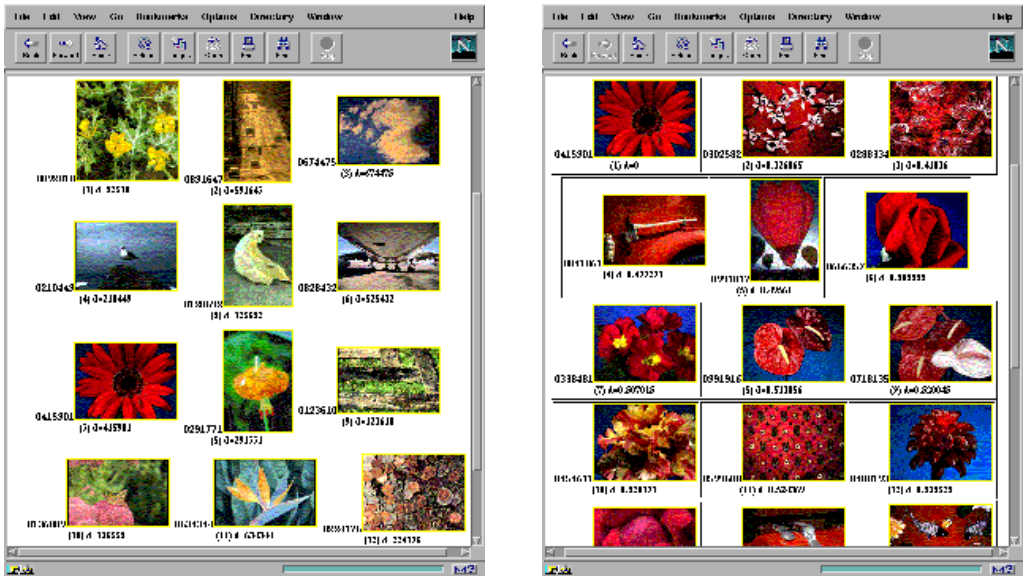
8.3.2 Relevance feedback queries

After the system finds the matches and returns the thumbnail images, as illustrated in Figure 8-11(b), the user has several options: (1) select images for full retrieval and viewing, (2) modify the query and try again and (3) select images for relevance feedback queries. In a relevance feedback query, VisualSEEk searches the database for the best matches to the selected image from the prior query results. The relevance feedback query tools allow the user to select from several global similarity metrics such as: global color, color regions, global texture, and joint color and texture. Figure 8-12 illustrates an example relevance feedback query using global color features of a selected image from Figure 6(a). The images displayed in Figure 8-12(b) have the most similar color content to selected image.



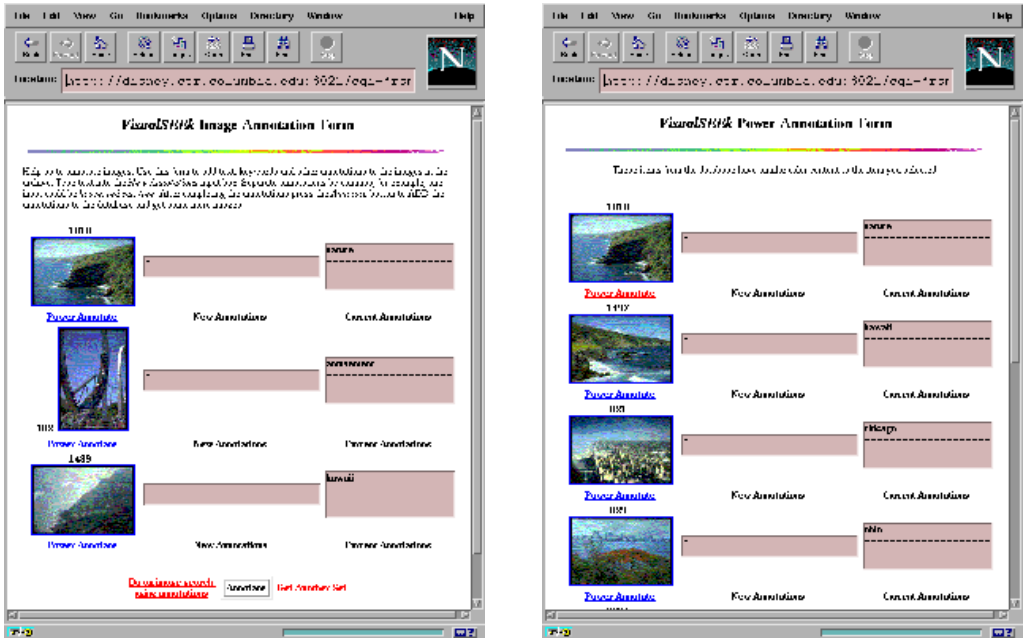
(a.) (b.)

Figure 8-11: An example VisualSEEk query that specifies (a) three color regions - blue, green and light blue, and their locations and sizes, and (b) the query results.



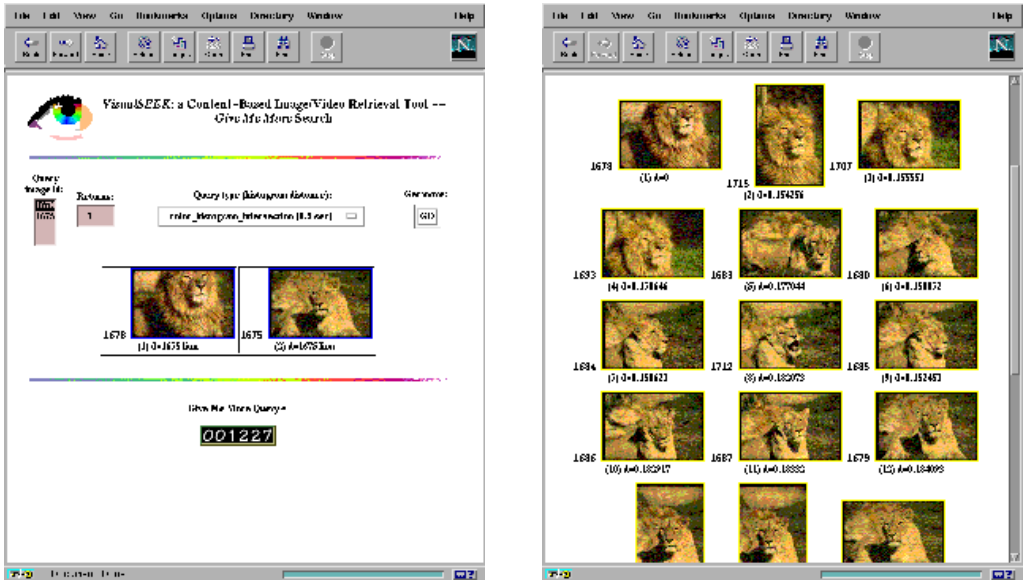
(a.) (b.)

Figure 8-12: Relevance feedback query. (a) Images selected at random from the database, and (b) the results of a color-query using the image in (a) third down on the left.



(a.) (b.)

Figure 8-13: Image annotation process. (a) Individual image annotation. (b) "Power" annotation allows groups of similar images to be annotated at once.



(a.) (b.)

Figure 8-14: Text and color-searching. (a) The text-based search using keyword "lion" yields only two matches. (b) The color search using one lion image from (a) finds many more images of lions.

8.3.3 Image annotation

The VisualSEEk system also provides tools for annotating images individually and in groups. The annotation interface, illustrated in Figure 8-13(a), allows the user to assign text to the images. The user adds new annotations by entering keywords into the entry boxes corresponding to the images. VisualSEEk also provides a system for annotating the images in groups.

By clustering images from the database by color, the user annotates them in groups, which significantly reduces the time and effort required to annotate the archive. For example, the first image, which depicts an ocean scene, see Figure 8-13(a), is selected for "power" annotation. The system searches the database for similar images and returns them to the user. Many of the returned images have a similar content to the selected image (see in Figure 8-13 that several of these depict ocean scenes). The user now assigns annotations, as appropriate, to the group of images, thereby speeding-up the process of annotating the images in the database.

8.3.4 Text-based searching

Text-based searching is still an important component of the image retrieval systems. Although text alone is only partially for image retrieval, text-searching combined with content-based techniques improves query capabilities. For example, given that some of the images are annotated, a user finds some matches in a text-based search. If the text-based search is partially successful, the user selects some of the matches to seed the relevance feedback queries.

For example, say that only two of the many images of lions from the database have been annotated with the term "lion." By conducting a text-based search using the term "lion," the system finds only these two images, as illustrated in Figure 8-14(a). However, many images of lions remain in the archive and are not retrieved because they have not been annotated with the term "lion." By selecting one of the returned lion images and by issuing a relevance feedback query, the system finds many of the remaining images of lions, as illustrated in Figure 8-14(b).

8.3.5 Color/spatial image query

The VisualSEEk system provides many useful components of an image retrieval system: the color/spatial tools, relevance feedback search system, "power annotation" capability and text-based searching. However, the color/spatial query system is designed specifically for querying by the locations, sizes and colors of image regions. In order to extend the integrated spatial and feature image query paradigm to other features, such as texture, and to incorporate more powerful spatial indexing techniques, we developed the SaFe image query system.

8.4. SaFe spatial and feature query system

We briefly describe the SaFe system for querying for images by spatial and feature attributes. The SaFe spatial and feature image query system integrates content-based techniques with spatial querying in order to search for images by arrangements and visual features of regions.

8.4.1 System design

The system is composed of four sub-systems: (1) the image analysis subsystem, (2) the query engine, (3) the database and (4) the user-interface, as illustrated in Figure 8-15. These sub-systems are combined to provide a complete solution for the search and retrieval of images.

8.4.2 Image query process

The spatial and feature image query process provides a powerful tool for image retrieval. However, it is extremely complex in that it requires that several disparate image query techniques be seamlessly integrated. First, the SaFe query system requires the assessment of feature similarities. Content-based systems, such as those proposed by [102, 3, 159], provide techniques for comparing visual features such as color, texture and shape. In Chapter 4, we explored several techniques for querying by visual features which, which are used in SaFe to compare the visual features of images and regions.

Second, the SaFe query system requires the assessment of the similarities in spatial locations and sizes. Third, the system requires that the spatial relationships, such as "above" and "below," can be resolved.

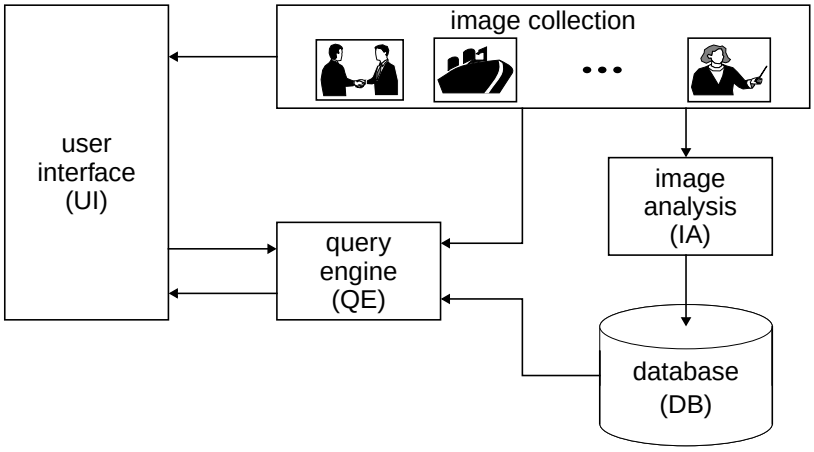


Figure 8-15: SaFe system architecture. Major subsystems are UI = user-interface, IA = image analysis subsystem, QE = query engine and DB = image database.

Finally, the system requires that all of these separate query tools are integrated into a single system. We investigated these systems for spatial querying and integrated spatial and feature image querying in detail in Section 7.

8.4.3 SaFe image query examples

In the SaFe system, the spatial queries are formulated graphically, as illustrated in Figure 8-16. The user sketches image regions, positions them on the query grid and assigns them properties of color, size and absolute location. The user may also assign boundaries for location and size. For example, in Figure 8-16, the query consists of two color regions - light blue and purple, which are positioned on the query grid. The query is constrained to return images that contain regions with colors similar to these, and in similar locations. The relative spatial constraint is not selected for this query.

The matches are illustrated from left to right, and from top to bottom, in order of closest distance to the query image. These synthetic images were used as a color-region image test-set in order to evaluate the retrieval effectiveness of the SaFe image query system in Section 7.

The SaFe system also provides a powerful platform for retrieving color photographic images, as illustrated in Figure 8-17. In this example, the user specifies two color regions - top is blue and bottom is green. The best matching images contain blue and green regions in locations similar to those depicted on the query grid. We demonstrated in Section 7.4.6 that the SaFe queries for "unconstrained" photographic color images improve retrieval effectiveness significantly over previous content-based techniques.

8.5. Summary

We introduced three new prototype systems that demonstrate the techniques for image retrieval developed in this thesis. The WebSEEk system catalogs and indexes the visual information on the Web [147]. WebSEEk is an advanced content-based image and video search engine for the World-Wide Web. The system automatically collects the images and videos and catalogs them using both the textual and visual information. WebSEEk is very easy to use and provides great flexibility and functionality for searching and browsing for images and videos on the Web. In the initial implementation, the system has catalogued more than one half million images and videos.

Because the discrimination of images is only partly provided by global features such as color histograms, we developed the VisualSEEk and SaFe image systems. VisualSEEk extracts and indexes the single color region from images [146]. The user searches for color images by specifying the locations of the color regions. VisualSEEk also provides systems for clustering and annotating images, and for iterating the query process through relevance feedback queries.

Finally, SaFe is an integrated spatial and feature image query system. SaFe utilizes image regions and their features, sizes, spatial locations, and relationships in order to compare images [152]. This integration of

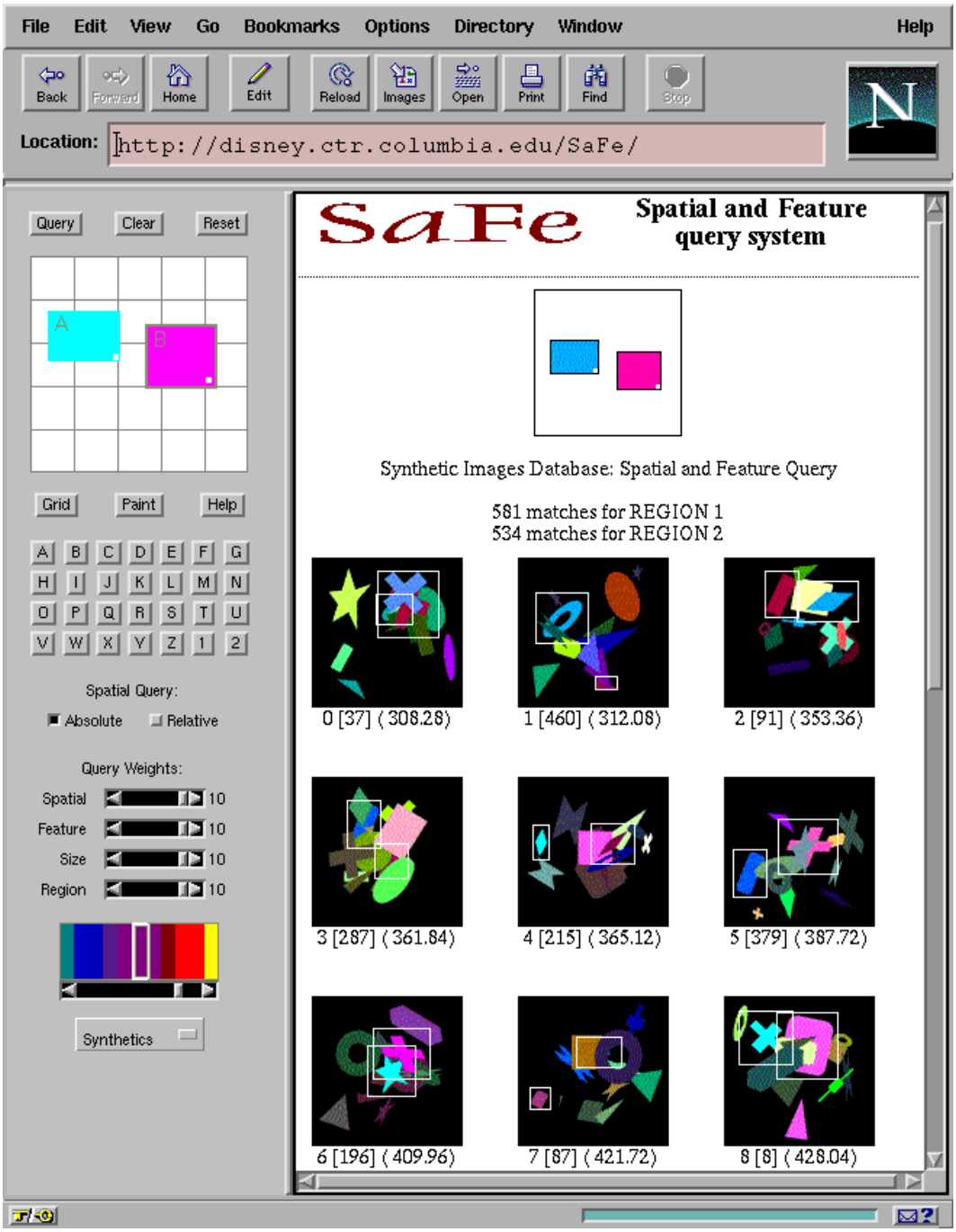


Figure 8-16: SaFe image retrieval of synthetic color-region images.

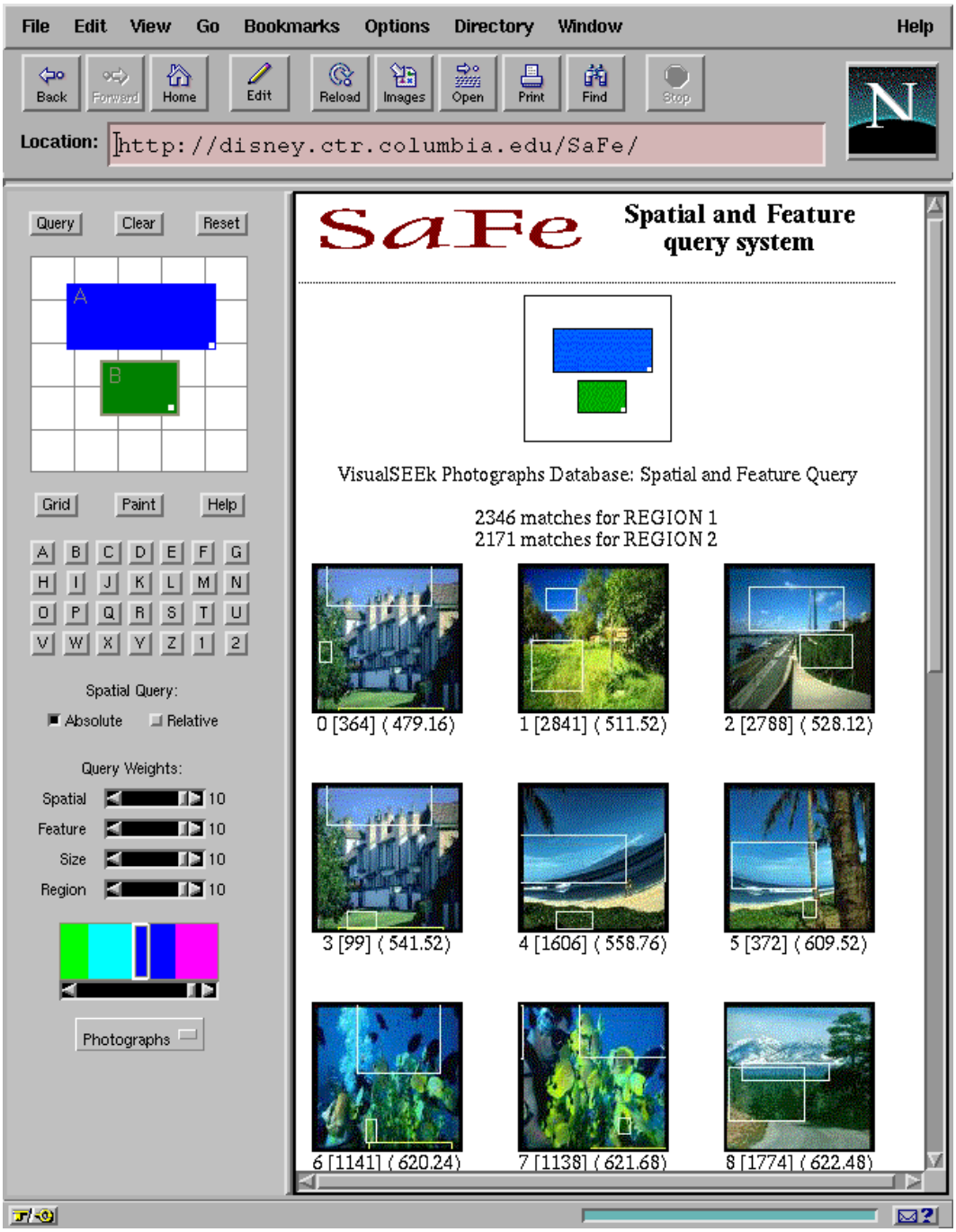


Figure 8-17: SaFe image retrieval of "unconstrained" color photographs.

content-based and spatial querying provides for the highly functional image query systems which allow for a wide variety of integrated spatial and feature queries.

Chapter 9

Conclusion and future directions

9.1. Conclusion

This thesis developed several new techniques for integrating spatial and feature information in order to improve systems for image retrieval, analysis and compression. The visually important information within images is often contained within spatially localized regions, or is represented by the overall existence and spatial arrangements of these regions. By developing processes that analyze and represent images in this way, we improve our capabilities to develop powerful content-based image retrieval and compression systems. This thesis presented several new techniques for integrating spatial and feature image analyses and representations, and demonstrated improved performance in image systems applications.

Recent efforts in image retrieval applications have focused on indexing a few specific visual dimensions of the images, such as color, texture, shape, motion and spatial information [102, 3, 112, 105, 156, 146]. However, without integrating these visual dimensions, the current content-based techniques have limited capacity to satisfactorily retrieve images.

Similarly, recent efforts in image compression have designed techniques for adapting to either the image frequency spectrum [32, 170, 124] or spatial information [136, 155, 132] in order to compact the images. However, by integrating spatial and frequency methods, we design algorithms which better adapt to the image content and improve compression.

This thesis developed new techniques to solve these problems which are similar in their exploitation of images as two-dimensional, non-stationary signals. In particular, we developed new processes that

1. **Image retrieval** – demonstrated a new integrated spatial and feature query paradigm for image retrieval. Previous efforts in image database systems have developed either feature-based approaches [159, 53, 102, 157, 51] or spatial-symbolic image retrieval methods [55, 22, 27, 113]. This thesis developed a new method which represents images by spatially localized regions. The features (color and texture) and spatial attributes (region size and location) define the set of properties of the regions. The images are compared by assessing the numbers of-, arrangements and feature properties of regions. We presented techniques for computing these complex image queries and demonstrated improved effectiveness in image retrieval applications.

2. **Image analysis** – We developed a framework for extracting spatially localized features from images using histogram and binary set back-projections. The automated extraction of regions from "unconstrained" imagery is not successfully accomplished using current methods of pattern recognition [133, 40] and image segmentation [155, 67, 73, 86, 90, 14]. The difficulty results from the inability to define either the complete set of region models or patterns *a priori*, or establish the ground-truth segmentations for "unconstrained" images. We developed a new approach for extracting color and texture regions by back-projecting the feature-histograms onto the images. We also developed a process by which the regions are extracted automatically by testing back-projections of binary feature sets. This approach represents a departure from traditional "constrained-domain" methods, however, we demonstrated that when combined with the integrated spatial and feature query paradigm, a powerful automated image indexing and retrieval system is developed.

3. **Image compression** – We developed a new system for compressing and analyzing images by decomposing the images into spatially and frequency localized basis elements. We developed a theory of "partitionable" frequency expansions which enable spatial segmentation and frequency expansion to be

integrated into a single graph-structured decomposition of the images [144, 150]. The joint adaptive space and frequency (JASF) graph improves image compression performance over previous adaptive image expansion techniques (such as those in [62, 32, 124]).

This thesis also developed a new representation of texture based upon texture channel energy histograms. Unlike other representations of texture [30, 89, 46, 60, 74, 84], the texture histograms allow the image indexing techniques developed for color histograms to be compatible with the representation of texture.

We also explored representations of color and texture using binary feature sets. We demonstrated advantages in efficiency in computing feature similarities, region extraction and integrated spatial and feature image retrieval. We presented a new approach for bounding histogram queries (BSB) by computing instead binary set queries. We presented another efficient approach for computing feature similarity queries using a query-optimized distance (QOD) computation. We demonstrated that these techniques increase the efficiency of feature-based image retrieval systems.

Finally, several new prototype image systems have been developed for this thesis – WebSEEk, VisualSEEk and SaFe, to provide test-beds for exploring new methods of image retrieval. These prototype systems demonstrate the proposed techniques for integrating spatial and feature techniques for images developed in this thesis. They also demonstrate other interesting techniques for automatically cataloging images and videos, content-based relevance feedback searching, "power" annotation, and integrated text and content-based image searching.

9.2. Future directions

Many difficult problems in image systems remain to be investigated. The explosive proliferation of "unconstrained" digital imagery in the form of images, graphics, and videos, due to the improved accessibility of computer technology and the main-stream acceptance of the World-Wide Web as a viable medium for publishing, advertising and communicating has created an immediate need for efficient and effective tools for cataloging, indexing, managing, compressing and searching for the "unconstrained" visual information. A significant gap remains between the ability of computers to analyze images and videos at the feature-level (colors, textures, shapes) compared to the inability at the semantic-level (objects, scenes, people, moods, artistic value). Closing this gap by improving technologies for image understanding would certainly improve the image search and retrieval systems.

However, the solution of this "large" problem (involving vision, cognition, subjectivity) remains distant while much of the current technology consists of solutions to "small" problems (for example, image segmentation [111], color constancy [65], shape from texture [4], video shot detection [2], and so forth).

However, there are new opportunities to climb the semantic ladder in today's diverse media environments. For example, as demonstrated in a small capacity by the WebSEEk system [148], the World-Wide Web facilitates the intersection of the previous separate technologies of natural language processing, automated image analysis, learning systems and information retrieval. There is great hope that at these new points of "intersection" of diverse technologies and application environments, new solutions to the problems of visual information management will be found.

9.3. Afterword

As a final note, we outline one possible step in this direction, which develops a system for decoding image semantics from color regions. In this problem we first define and annotate a set of region templates, which consist of regions with spatial and feature properties. These may be obtained by learning from interaction with users in the process of image retrieval, or be generated manually to provide an object database. We also define an image template dictionary, which assigns names to images that are defined by arrangement of regions. These can be developed in a variety of ways in different applications, for example, from textual information in the closed captioning of broadcast video, from text on Web pages surrounding the images, or from embedded text within images.

Given these pieces, we design an algorithm which automatically chooses the best image name for an unknown image by decomposing the image into spatially localized region. The algorithm picks the set of region templates that most likely matches those in the unknown image, given the constraints of the image template dictionary. We suggest that the Viterbi algorithm provides one possible tool for solving problems that are formulated in this framework.

References

[1] R. Agarwal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms (FODO) Conference*, Evanston, Ill, October 1993.

[2] G. Ahanger and T. D. Little. A survey of technologies for parsing and indexing digital video. *Journ. Visual Comm. and Image Rep.*, 7(1):28 – 43, March 1996.

[3] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. C. Jain, and C. Shu. Virage image search engine: an open framework for image management. In *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, volume 2670, pages 76 – 87. IS&T/SPIE, 1996.

[4] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Inc, Englewood Cliffs, NJ, 1982.

[5] J. Beck, A. Sutter, and R. Ivry. Spatial frequency channels and perceptual grouping in texture segregation. *Computer Vision, Graphics and Image Processing (CVGIP)*, 37, 1987.

[6] F. Bergeaud and S. Mallat. Matching pursuit of images. In *IEEE Proc. Int. Conf. Acoust., Speech, Signal Processing*, May 1995.

[7] P. V. Biron and D. H. Kraft. New methods for relevance feedback: Improving information retrieval performance. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, pages 482 – 487, 1995.

[8] A. C. Bovick, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(1), January 1990.

[9] P. Brodatz. *Textures; a photographic album for artists and designers*. Dover Publications, Inc., New York, NY, 1966.

[10] J. M. H. Du Buf. Abstract process in texture discrimination. *Spatial Vision*, vol 6 1992.

[11] J. M. H. Du Buf, M. Kardan, and W. Spann. Texture feature performance for image segmentation. *Pattern Recog.*, 23(3/4), 1990.

[12] T. Caelli and D. Reye. On the classification of image regions by colour, texture and shape. *Pattern Recog.*, 26(4), 1993.

[13] C. Cedras and M. Shah. Motion-based recognition: a survey. *Image and Vision Computing*, 13(2), March 1995.

[14] M. Celenk. A color clustering technique for image segmentation. *Computer Vision, Graphics and Image Processing (CVGIP)*, vol. 52 1990.

[15] N. S. Chang and K. S. Fu. Query-by-pictorial-example. *IEEE Trans. Softw. Engin.*, November 1980.

[16] N. S. Chang and K. S. Fu. Picture query languages for pictorial data-base systems. *IEEE Computer*, November 1981.

[17] S.-F. Chang. Compressed-domain techniques for image/video indexing and manipulation. In *Proceedings, I.E.E.E. International Conference on Image Processing*, Washington, DC, Oct. 1995. invited paper to the special session on Digital Library and Video on Demand.

[18] S.-F. Chang. Compressed-domain techniques for image/video indexing and manipulation. In *IEEE Proc. Int. Conf. Image Processing*, Special Session on Digital Library and Video-On Demand, Washington, DC, October 1995. IEEE.

[19] S.-F. Chang, A. Eleftheriadis, and D. Anastassiou. Development of columbia's video on demand testbed. *Image Communication*, 1996.

[20] S.-K. Chang. *Principles of Pictorial Information Systems Design*. Prentice-Hall, Inc, Englewood Cliffs, NJ, 1989.

[21] S.-K. Chang and T. L. Kunii. Pictorial data-base systems. *IEEE Computer*, November 1981.

[22] S.-K. Chang, Q. Y. Shi, and C. Y. Yan. Iconic indexing by 2-D strings. *IEEE Trans. Pattern Anal. Machine Intell.*, 9(3):413 – 428, May 1987.

[23] T. Chang and C.-C. Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Trans. Image Processing*, 3(4), October 1993.

[24] B. B. Chaudhari. and N. Sarkar. Texture segmentation using fractal dimension. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(1), January 1995.

[25] R. Chellappa, R. L. Kashyap, and B. S. Manjunath. Model-based texture segmentation and classification. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, pages 277 – 310. World Scientific Publishing Company, River Edge, NJ, 1993.

[26] J.-L. Chen and A. Kundu. Rotation and gray scale transform invariant texture identification using wavelet decomposition and hidden markov model. *IEEE Trans. Pattern Anal. Machine Intell.*, 16(2), February 1994.

[27] S.-S. Chen. Content-based indexing of spatial objects in digital libraries. *Journ. Visual Comm. and Image Rep.*, 7(1):16 – 27, March 1996.

[28] M. Chock, A. F. Cardenas, and A. Klinger. Manipulating data structures in pictorial information systems, November 1981.

[29] T.-S. Chua, S.-K. Lim, and H.-K. Pung. Content-based retrieval of segmented images. In *Proc. ACM Intern. Conf. Multimedia*, October 1994.

[30] F. S. Cohen, Z. Fan, and M. A. Patel. Classification of rotated and scaled textured images using guassian markov random field models. *IEEE Trans. Pattern Anal. Machine Intell.*, 13(2), February 1991.

[31] R. R. Coifman, Y. Meyer, S. Quake, and M. V. Wickerhauser. Signal processing and compression with wave packets. Technical report, Yale University Department of Math technical report, April 1990.

[32] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Trans. Inform. Theory*, 38(2), March 1992.

[33] S. Cooper. *Find Waldo Now.* Little, Brown and Company, 1988.

[34] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., New York, NY, 1991.

[35] J. G. Daugman. Complete discrete 2-D gabor transform by neural networks for image analysis and compression. *IEEE Trans. Acoust., Speech, Signal Processing*, 36(7), July 1988.

[36] J. G. Daugman. Entropy reduction and decorrelation in visual coding by oriented neural receptive fields. *IEEE Trans. Biomed. Engin.*, 36(1), January 1989.

[37] M. Davis. Media streams: An iconic language for video annotation. *Telektronik 4.93: Cyberspace*, 1993.

[38] D. Deloddere, W. Verbiest, and H. Verhille. Interactive video on demand, May 1994.

[39] W. R. Dillon and M. Goldstein. *Multivariate analysis : methods and applications.* John Wiley & Sons, Inc., 1984.

[40] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, Inc., 1973.

[41] D. Dunn and W. E. Higgins. Optimal gabor filters for texture segmentation. *IEEE Trans. Image Processing*, 4(7), July 1995.

[42] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, New York, NY, 1994.

[43] F. Ennesser and G. Medioni. Finding waldo, or focus of attention using local color information. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(8), August 1995.

[44] C. Faloutsos, R. Barber, M. Flickner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3, 1994.

[45] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23 – 32, September 1995.

[46] J. M. Francos, A. Z. Meiri, and B. Porat. A unified texture model based on a 2-D Wold like decomposition. Technical report, Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel, 1995.

[47] C. Frankel, M. Swain, and V. Athitsos. Webseer: An image search engine for the world wide web. Technical Report TR-96-14, University of Chicago, July 1996. `http://webseer.cs.uchicago.edu/`.

[48] K. Fukunaga. *Introduction to Statistical Pattern Recogntion.* Harcourt Brace Javanovich, 1990.

[49] D. Le Gall. MPEG: A video compression standard for multimedia applications. *Commun. ACM*, 34(4), April 1991.

[50] A. Gersho and R. M. Gray. *Vector quantization and signal compression.* Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 1992.

[51] M. M. Gorkani and R. W. Picard. Texture orientation for sorting photos 'at a glance'. In *IEEE Conference on Pattern Recognition*, October 1994. MIT Technical Report 292.

[52] C. C. Gotlieb and H. E. Kreyszig. Texture descriptors based on co-occurrence matrices. *Computer Vision, Graphics and Image Processing (CVGIP)*, vol. 51 1990.

[53] R. S. Gray. Content-based image retrieval: Color and edges. Technical report, Dartmouth University Department of Computer Science technical report #95-252, 1995.

[54] E. Griffiths and T. Troscianko. Can human texture discrimination be mimicked by a computer model using local fourier analysis. *Spatial Vision*, vol. 6 1992.

[55] V. N. Gudivada and V. V. Raghavan. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Trans. on Information Systems*, 13(2), April 1995.

[56] O. Guenther and A. Buchmann. Research issues in spatial databases. In *ACM SIGMOD Record*, volume 19, December 1990.

[57] S. R. Gunn and M. S. Nixon. A robust snake implementation; a dual active contour. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(1):63 – 68, January 1997.

[58] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM Proc. Int. Conf. Manag. Data (SIGMOD)*, pages 47 – 57, June 1984.

[59] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. Pattern Anal. Machine Intell.*, July 1995.

[60] R. M. Haralick. Statistical and structural approaches to texture. *Proc. of the IEEE*, 67(5), May 1979.

[61] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Trans. Syst., Man, Cybern.*, smc-3(6), November 1973.

[62] C. Herley, J. Kovačević, K. Ramchandran, and M. Vetterli. Tilings of the time-frequency plane: Constructions of arbitrary orthogonal bases and fast tiling algorithms. *IEEE Trans. Signal Processing*, December 1993.

[63] C. Herley, Z. Xiong, K. Ramchandran, and M. T. Orchard. An efficient algorithm to find a jointly optimal time-frequency segmentation using time-varying filter banks. In *IEEE Proc. Int. Conf. Acoust., Speech, Signal Processing*, May 1995.

[64] G. R. Hofmann. The modelling of images for communication in multimedia environments and the evolution from the image signal to the image document. *The Visual Computer*, 9, 1993.

[65] B. Klaus Paul Horn. *Robot Vision*. The MIT Electrical Engineering and COmputer Science Series. The MIT Press, Cambridge, MA, 1986.

[66] W. Hsu, T. S. Chua, and H. K. Pung. An integrated color-spatial approach to content-based image retrieval. In *Proc. ACM Intern. Conf. Multimedia*, November 1995.

[67] Y. Hu and T. J. Dennis. Textured image segmentation by context enhance clustering. *IEE Pro.-Vis. Image Signal Process.*, 1994.

[68] R. W. G. Hunt. *Measuring Color*. Ellis Horwood series in applied science and industrial technology. Halsted Press, New York, NY, 1989.

[69] G. Hunter and K. Steiglitz. Operations on images using quad trees. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-1(2), April 1979.

[70] Interpix. Image surfer. `http://www.interpix.com/`.

[71] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series*, pages 277 – 286, Los Angeles, CA, 1995.

[72] H. V. Jagadish. A retrieval technique for similar shapes. In *ACM Proc. Int. Conf. Manag. Data (SIGMOD)*, February 1991.

[73] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recog.*, 24(12), 1991.

[74] M. E. Jernigan and F. D'Astous. Entropy-based texture analysis in the spatial frequency domain. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6(2), March 1984.

[75] J. D. Johnston. A filter family desgined for use in quadrature mirror filter banks. In *IEEE Proc. Int. Conf. Acoust., Speech, Signal Processing*, pages 291 – 294, April 1980.

[76] K. Sparck Jones. *Information Retrieval Experiment.* Butterworth and Co., Boston, 1981.

[77] J. J. Rocchio Jr. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313 – 323. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[78] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321 – 331, 1988.

[79] P. M. Kelly and T. M. Cannon. Candid: Comparison algorithm for navigating digital image databases. In *IEEE 7th International Working Conference on Scientific and Statistical Database Management*, September 1994.

[80] B. Kernighan and D. Ritchie. *The C Programming Language.* Prentice Hall Software Series. Prentice-Hall, Inc, Englewood Cliffs, NJ, 2 edition, 1988.

[81] A. Khotanzad and J.-Y. Chen. Unsupervised segmentation of textured images by edge selection in multidimensional features. *IEEE Trans. Pattern Anal. Machine Intell.*, 11(4), April 1989.

[82] T. Kirste. SPACEPICTURE - an interactive hypermedia satellite image archival system. *Computers & Graphics*, 17(3), 1993.

[83] F. Korn, N. Sidiropoulos, E. Siegel, and Z. Protopapas. Fast nearest neighbor search in medical image databases. Technical Report 3613, Department of Computer Science, University of Maryland, 1995.

[84] A. Kundu and J.-L. Chen. Texture classification using QMF bank-based subband decomposition. *Computer Vision, Graphics and Image Processing (CVGIP): Graphical Models and Image Processing*, 54(5), September 1992.

[85] K. Y. Kupeev and H. J. Wolfson. On shape similarity. Technical Report 293/94, Computer Science Department, Tel Aviv University, 1994.

[86] A. Laine and J. Fan. An adaptive approach for texture segmentation by multi-channel wavelet frames. Technical Report TR-93-025, Center for Computer Vision and Visualization, August 1993.

[87] C.-S. Li and J. Turek. Content-based indexing of earth observing satellite image database with fuzzy attributes. In *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, volume 2670, pages 438 – 449. IS&T/SPIE, 1996.

[88] T. D. C. Little, G. Ahanger, R. J. Folz, J. F. Gibbon, F. W. Reeve, D. H. Schelleng, and J. F. Venkatesh. A digital on-demand video service supporting content-based queries. In *Proc. ACM Intern. Conf. Multimedia*, June 1993.

[89] F. Liu and R. W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. Technical Report 320, MIT Media Laboratory and Modeling Group Technical Report, 1994.

[90] J. Liu and Y.-H. Yang. Multiresolution color image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 16(7), December 1994.

[91] Q.-T. Luong. Color in computer vision. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, pages 311 – 368. World Scientific Publishing Company, River Edge, NJ, 1993.

[92] W. Y. Ma and B. S. Manjunath. Texture-based pattern retrieval from image databases. *Multimedia Tools and Applications*, 1(2):35 – 51, 1996.

[93] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Machine Intell.*, 11(7), July 1989.

[94] S. Mallat and Z. Zhang. Matching pursuit with time-frequency dictionaries. *IEEE Trans. Signal Processing*, December 1993.

[95] J. Meng and S.-F. Chang. CVEPS - a compressed video editing and parsing system. In *Proc. ACM Intern. Conf. Multimedia*, pages 43 – 53, Boston, MA, November 1996. ACM. `http://www.ctr.columbia.edu/cveps/`.

[96] J. Meng and S.-F. Chang. Tools for compressed-domain video indexing and editing. In *IS&T/SPIE Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, San Jose, CA, February 1996.

[97] T. P. Minka and R. W. Picard. An image database browser that learns from user interaction. Technical Report 365, MIT Media Laboratory and Modeling Group Technical Report, 1996.

[98] M. Miyahara and Y. Yoshida. Mathematical transform of (r, g, b) color data to Munsell (h, v, c) color data. In *SPIE Visual Communications and Image Processing 1988*, vol. 1001 1988.

[99] A. H. Munsell. *A Color Notation: An Illustrated System Defining All Colors and Their Relations By Measured Scales of Hue, Value and Saturation.* Munsell Color Company, 1905.

[100] A. H. Munsell. *Munsell Book of Color.* Munsell Color Company, 1942.

[101] A. N. Netravali and B. G. Haskell. *Digital Pictures; representation and compression.* Applications of Communications Theory. Plenum Press, New York, NY, 1988.

[102] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, and C. Faloutsos. The QBIC project: Querying images by content using color, texture, and shape. In *Storage and Retrieval for Image and Video Databases*, volume SPIE Vol. 1908, February 1993.

[103] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, and C. Faloutsos. The QBIC project: Querying images by content using color, texture, and shape. *IBM RJ 9203 (81511)*, February 1993.

[104] H. Nieman. *Pattern Analysis and Understanding.* Springer Series in Information Sciences. Springer-Verlag, 2nd ed. edition, 1990.

[105] V. E. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40 – 48, September 1995.

[106] P. P. Ohanian and R. C. Dubes. Performance evaluation for four classes of textural features. *Pattern Recog.*, 25(8), 1992.

[107] Y.-I. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, 13:222 – 241, 1980.

[108] S. C. Orphanoudakis, C. Chronaki, and S. Kostomanolakis. I2c: A system for the indexing, storage, and retrieval of medical images by content. Technical report, Technical report - University of Crete, January 1994.

[109] A. Ortega, Z. Zhang, and M. Vetterli. Modeling and optimization of a multiresolution remote image retrieval system. In *INFOCOM 1994*, 1994.

[110] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *Proc. ACM Intern. Conf. Multimedia*, Boston, MA, 1996.

[111] T. Pavlidis. *Algorithms for Graphics and Image Processing.* Computer Science Press, Inc., 1982.

[112] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *Proceedings of the SPIE Storage and Retrieval Image and Video Databases II*, February 1994.

[113] E. G. M. Petrakis. Image representation, indexing and retrieval based on spatial relationships and properties of objects, March 1993.

[114] E. G. M. Petrakis and C. Faloutsos. Similarity searching in large image databases. Technical Report 3388, Department of Computer Science, University of Maryland, 1995.

[115] R. W. Picard. Content access for image/video coding 'the fourth criterion'. Technical Report 295, MIT Media Laboratory and Modeling Group Technical Report, October 1994.

[116] R. W. Picard and T. Kabir. The Brodatz texture database: Characterization by shift-invariant principal components. Technical Report 203, MIT Media Laboratory and Perceptual Computing Group Technical Report, 1993.

[117] R. W. Picard and T. Kabir. Finding similar patterns in large image databases. Technical Report 205, MIT Media Laboratory and Modeling Group Technical Report, April 1993.

[118] R. W. Picard, T. Kabir, and F. Liu. Real-time recognition with the entire Brodatz texture database. Technical Report 215, MIT Media Laboratory and Perceptual Computing Group Technical Report, June 1993.

[119] R. W. Picard and F. Liu. A new wold ordering for image similarity. Technical Report 237, MIT Media Laboratory and Modeling Group Technical Report, 1993.

[120] M. Porat and Y. Y. Zeevi. The generalized gabor scheme of image representation in biological and machine vision. *IEEE Trans. Pattern Anal. Machine Intell.*, 10(4), July 1988.

[121] M. Porat and Y. Y. Zeevi. Localized texture processing in vision: Analysis and synthesis in the gaborian space. *IEEE Trans. Biomed. Engin.*, January 1989.

[122] K. Price. Image segmentation: A comment on 'studies in global and local histogram-guided relaxation algorithms'. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6(2), March 1984.

[123] K. Ramchandran, A. Ortega, and M. Vetterli. Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders. *IEEE Trans. Image Processing*, April 1993.

[124] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Trans. Image Processing*, June 1993.

[125] A. Rao, Ravishanka, and G. L. Lohse. Identifying high level features of texture perception. *Computer Vision, Graphics and Image Processing (CVGIP): Graphical Models and Image Processing*, 1993.

[126] K. R. Rao and J. J. Hwang. *Techniques and Standards for Image, Video, and Audio Coding.* Prentice-Hall, Inc, 1996.

[127] T. Reed and J. M. H. Du Buf. A review of recent texture segmentation feature extraction techniques. *Computer Vision, Graphics and Image Processing (CVGIP): Image Understanding*, 57(3), May 1993.

[128] T. Reed and H. Wechsler. Segmentation of textured images and gestalt organization using spatial/spatial-frequency representations. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(1), January 1990.

[129] A. R. Robertson. Historical development of CIE recommended color difference equations. *COLOR Research and Application*, 15(3), June 1990.

[130] L. A. Rowe and R. R. Larson. A video-on-demand system. Technical report, UC Berkeley technical report, 1993.

[131] J. C. Russ. *The Image Processing Handbook.* CRC Press, Boca Raton, 1995.

[132] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187 – 260, 1984.

[133] J. Schurmann. *Pattern classification : A Unified View of Statistical and Neural Approaches.* John Wiley & Sons, Inc., 1996.

[134] S. Sclaroff. World wide web image search engines. In *NSF Workshop on Visual Information Management*, Cambridge, MA, June 1995.

[135] S. Sclaroff and A. P. Pentland. On modal modeling for medical images: Underconstrained shape description and data compression. In *IEEE Workshop on Biomedical Image Analysis*, Seattle, WA, 1994.

[136] E. Shusterman and M. Feder. Image compression via improved quadtree decomposition algorithms. *IEEE Trans. Image Processing*, 3(2):207 – 215, 1994.

[137] E. P. Simoncelli. Distributed representation and analysis of visual motion. Technical Report 209, MIT Media Laboratory and Modeling Group Technical Report, January 1993. Ph.D. Dissertation.

[138] J. R. Smith and S.-F. Chang. Quadtree segmentation for texture-based image query. In *Proc. ACM Intern. Conf. Multimedia*, pages 279 – 286, San Francisco, CA, October 1994. ACM.

[139] J. R. Smith and S.-F. Chang. Transform features for texture classification and discrimination in large image databases. In *Proc. Int. Conf. Image Processing*, pages 407 – 411, Austin, TX, November 1994. IEEE.

[140] J. R. Smith and S.-F. Chang. Frequency and spatially adaptive wavelet packets. In *Proc. Int. Conf. Acoust., Speech, Signal Processing*, pages 2233 – 2236, Detroit, MI, May 1995. IEEE.

[141] J. R. Smith and S.-F. Chang. Automated binary texture feature sets for image retrieval. In *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, GA, May 1996. IEEE.

[142] J. R. Smith and S.-F. Chang. Integrated spatial and feature image query. Technical report, Columbia University, December 1996.

[143] J. R. Smith and S.-F. Chang. Searching for images and videos on the World-Wide Web. Technical Report CU/CTR 459-96-25, Columbia University, August 1996. `http://www.ctr.columbia.edu/webseek`.

[144] J. R. Smith and S.-F. Chang. Space adaptive wavelet packet image compression. In *Symposium on Electronic Imaging: Science and Technology – Still Image Compression 1996*, volume 2669, pages 138 – 149, San Jose, CA, January 1996. IS&T/SPIE.

[145] J. R. Smith and S.-F. Chang. Tools and techniques for color image retrieval. In *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, volume 2670, pages 426 – 437, San Jose, CA, February 1996. IS&T/SPIE.

[146] J. R. Smith and S.-F. Chang. VisualSEEk: a fully automated content-based image query system. In *Proc. ACM Intern. Conf. Multimedia*, pages 87 – 98, Boston, MA, November 1996. ACM. `http://www.ctr.columbia.edu/VisualSEEk`.

[147] J. R. Smith and S.-F. Chang. Webseek: a content-based image and video search engine for the world-wide web. *To appear IEEE Multimedia*, August 1996.

[148] J. R. Smith and S.-F. Chang. An image and video search engine for the World-Wide Web. In *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases V*, San Jose, CA, February 1997. IS&T/SPIE.

[149] J. R. Smith and S.-F. Chang. Joint adaptive space and frequency graph. Technical report, Columbia University, January 1997.

[150] J. R. Smith and S.-F. Chang. Joint adaptive space and frequency graph basis selection. In *Proc. Int. Conf. Image Processing*. IEEE, June 1997. Submitted.

[151] J. R. Smith and S.-F. Chang. Querying by color regions using the *VisualSEEk* content-based visual query system. In M. T. Maybury, editor, *Intelligent Multimedia Information Retrieval*. AAA Press/MIT Press, Cambridge, MA, 1997.

[152] J. R. Smith and S.-F. Chang. Safe: A general framework for integrated spatial and feature image search. In *Workshop on Multimedia Signal Processing*, Princeton, NJ, June 1997. IEEE. Submitted.

[153] S. W. Smoliar and H. Zhang. Content-based video indexing and retrieval. *IEEE Multimedia Mag.*, Summer 1994.

[154] A. Soffer and H. Samet. Retrieval by content in symbolic-image databases. In *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, volume 2670, pages 144 – 155. IS&T/SPIE, 1996.

[155] A. Spann and R. Wilson. A quad-tree approach to image segmentation which combines statistical and spatial information. *Pattern Recog.*, 18(3/4), 1985.

[156] R. K. Srihari. Automatic indexing and content-based retrieval of captioned images. *IEEE Computer*, 28(9):49 – 56, September 1995.

[157] M. Stricker and A. Dimai. Color indexing with weak spatial constraints. In *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, volume 2670, pages 29 – 41. IS&T/SPIE, 1996.

[158] M. Stricker and M. Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases III*, volume SPIE Vol. 2420, February 1995.

[159] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7:1 1991.

[160] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Trans. Syst., Man, Cybern.*, SMC-8(6):460–473, 1978.

[161] H. Tamura and N. Yokoya. Image database systems: A survey. *Pattern Recog.*, 17(1), 1984.

[162] G. Tortora and G. Costagliola. Pyramidal algorithms for iconic indexing. *Computer Vision, Graphics and Image Processing (CVGIP)*, vol. 52 1990.

[163] H. Van Trees, editor. *Detection, Estimation, and Modulation Theory; part I, Detection, Estimation and Linear Modulation Theory.* John Wiley & Sons, Inc., New York, NY, 1968.

[164] M. Turceryan and A. Jain. Texture analysis. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, pages 235 – 276. World Scientific Publishing Company, River Edge, NJ, 1993.

[165] S. E. Umbaugh, R. H. Moss, and W. V. Stoecker. Automatic color segmentation of images with application to detection of variegated coloring in skin tumors, December 1989.

[166] M. Vetterli. Multi-dimensional sub-band coding: Some theory and algorithms. *Signal Processing*, pages 97 – 112, April 1984.

[167] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1995.

[168] G. K. Wallace. The JPEG still picture compression standard. *IEEE Trans. Consum. Electr.*, 38(1), February 1992.

[169] D. Weinshall, M. Werman, and A. Shashua. Shape tensors for efficient and learnable indexing. Technical report, Institute of Computer Science, The Hebrew University of Jerusalem, 1995.

[170] M. V. Wickerhauser. Picture compression by best-basis sub-band coding. Technical report, Yale University Department of Math technical report, January 1990.

[171] R. Wilson and G. H. Granlund. The uncertainty principle in image processing. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-6(6), November 1984.

[172] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: compressing and indexing documents and images.* Van Nostrand Reinhold, New York, NY, 1994.

[173] D. Woelk, W. Kim, and W. Luther. An object-oriented approach to multimedia databases. *ACM Databases*, 1986.

[174] J. W. Woods and Sean D. O'Neil. Subband coding of images. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-34(5), October 1986.

[175] G. Wyszecki and W. S. Stiles. *Color science : concepts and methods, quantitative data and formulae.* The Wiley series in pure and applied optics. John Wiley & Sons, Inc., New York, NY, 2nd ed. edition, 1982.

[176] J. You and H. A. Cohen. Classification and segmentation of rotated and scaled textured images using texture 'tuned' masks. *Pattern Recog.*, 26(2), 1993.

[177] D. Zhong, H. J. Zhang, and S.-F. Chang. Clustering methods for video browsing and annotation. In *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, volume 2670, San Jose, CA, February 1996. IS&T/SPIE.