UNIVERSITY of CALIFORNIA
Santa Barbara

Variational Image Segmentation and Curve Evolution on Natural Images

A dissertation submitted in partial satisfaction of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Baris Sumengen

September 2004

The dissertation of Baris Sumengen is approved.

Dr. Charles Kenney

Professor Shivkumar Chandrasekaran

Professor Kenneth Rose

Professor Eric Mjolsness

Professor B. S. Manjunath, Committee Chair

September 2004

Variational Image Segmentation and Curve Evolution on Natural Images

To my Father and Mother

for inspiring me and

making my education top priority of their lives.

Acknowledgements

The six years I spent at UC, Santa Barbara covers a significant portion of my life and has a larger meaning than the PhD title and the contents of this thesis for me. I consider this period as a “complete” experience where I learned, experienced and matured from many different aspects.

I lived the happiest period of my life during the time when my love, Ece Berkun, was in United States visiting me. I would like to thank Ece for her love and support during these years, which motivated me to improve myself significantly. I have had discussed many portions of this thesis with her and listened to her ideas. Section 5.7, where I introduce efficient implementations for Graph partitioning methods, inspired by an idea she mentioned at 7AM in the morning. I believe this section is one the key contributions of this thesis.

I would like to thank Professor Manjunath for making it possible for me to come to Santa Barbara for my graduate studies. He has been extremely supportive during these many years and I could not imagine a better advisor for my doctorate. While I argued with him about technical issues during the first couple of years, he ended up being right most of the time and I learned to trust his judgement. Professor Manjunath read this thesis many times and made countless corrections, which were very helpful for me to understand the technical issues and thesis writing in general.

Dr. Charles Kenney read parts of this thesis and made insightful comments. Professor Shiv Chandrasekaran was always available and helped me understand many technical and implementation issues. He pointed me in the right direction regarding 1) generating an edge function from Edgeflow vector field (Section 3.1) and 2) proof of Theorem 5.1.

My Colleague Dr. Jelena Tesic, who was also a PhD student during these six years, was very helpful in many ways. She helped me with several mathematical problems and took care of all the administrative issues and paperwork for me, which I usually avoid as much as possible. More importantly, she always helped me to put a smile on my face.

Dr. Shawn Newsam, again a graduate student during my studies, has been a good friend. We discussed countless ideas during the years. Section 6.1, category-based database design and image retrieval, is a joint project with him. He deserves at least half of the credit for this section. We also entered a business plan competition together using the ideas from Section 6.1, where we came fourth and earned a monetary price.

Other students in the Vision research lab have always been helpful and insightful. I have had extensive technical discussions with Sitaram Bhagavathy, Marco Zuliani and Motaz El-Saban, which helped me better understand many of the topics presented in this thesis. Former student of Vision Research Lab, Peng Wu, has helped me initially with a project related to MPEG-7 standardization, in which I participated right after I came to UCSB. Later on during my internship at HP Labs, he was my manager. He made my first industry experience very enjoyable for me. I would like to thank him for his support and friendship.

I would like to thank Samsung Electronics, the Office of Naval Research (Grants: ONR #N00014-01-1-0391 and ONR #N00014-04-1-0121), and the National Science Foundation (NSF ITR-0331697) for supporting this research.

Finally, I would like to thank my parents and my two sisters Dilek and Demet for their continuous love and endless support.

Disclaimer

This work was supported by grants from Samsung Electronics, the Office of Naval Research under grants: ONR #N00014-01-1-0391 and ONR #N00014-04-1-0121, and the National Science Foundation award NSF ITR-0331697.

Curriculum Vitæ

Baris Sumengen

June 1998	Bachelor of Science Department of Electrical and Electronic Engineering Bogazici University, Istanbul, Turkey
June 1998	Bachelor of Science Department of Mathematics Bogazici University, Istanbul, Turkey
March 2000	Master of Science Department of Electrical and Computer Engineering University of California, Santa Barbara
September 2004	Doctor of Philosophy Department of Electrical and Computer Engineering University of California, Santa Barbara

Fields of Study

Image processing, computer vision, pattern recognition, and data mining.

Publications

B. Sumengen, S. Bhagavathy, and B. S. Manjunath, “Graph Partitioning Active Contours for Knowledge-Based Geo-Spatial Segmentation,” *CVPR Workshop on Perceptual Organization in Computer Vision (POCV)*, 2004.

B. Sumengen, B. S. Manjunath, and C. Kenney, “Image segmentation using multi-region stability and edge strength,” *International Conference on Image Processing (ICIP)*, 2003.

B. Sumengen, B. S. Manjunath, and C. Kenney, “Image segmentation using curve evolution and flow fields,” *International Conference on Image Processing (ICIP)*, 2002.

B. Sumengen, B. S. Manjunath, and C. Kenney, “Image segmentation using curve evolution and region stability,” *International Conference on Pattern Recognition (ICPR)*, 2002.

S Newsam, B. Sumengen, B. S. Manjunath, “Category-based image retrieval,” *International Conference on Image Processing (ICIP)*, 2001.

B. Sumengen, B. S. Manjunath, and C. Kenney, “Image segmentation using curve evolution,” *Asilomar Conference on Signals, Systems and Computers*, 2001.

Abstract

Variational Image Segmentation and Curve Evolution on Natural Images

by

Baris Sumengen

The primary goal of this thesis is to develop robust image segmentation methods based on variational techniques. Image segmentation is one of the fundamental problems in image processing and computer vision. Segmentation is also one of the first steps in many image analysis tasks. Image understanding systems such as face or object recognition often assume that the objects of interests are well segmented. Different visual cues, such as color, texture, and motion, help in achieving segmentation. Segmentation is also goal dependent, subjective, and hence ill-posed in a general set up. However, it is desirable to consider generic criteria that can be applied to a large variety of images and can be adapted for specific applications. This thesis work focuses on developing such segmentation methods that work on natural images.

The first part of the dissertation proposes new designs for edge-based variational segmentation methods. Starting with the Edgeflow technique, which has been shown to be highly successful on natural images, two edge-based variational methods, a curve evolution method and an anisotropic diffusion method, are proposed. To verify the effectiveness of these new techniques, extensive tests are conducted on the Berkeley segmentation data set and associated ground truth.

The results show that our methods outperform the current state of the art. These methods are further extended to multi-scale image segmentation.

Second part of the dissertation explores region-based variational segmentation. We propose a new class of variational segmentation cost functions. Our cost functions are based on pair-wise dissimilarities between individual pixels and have been successfully applied to natural images by graph partitioning techniques. We minimize these cost functions within a variational framework. We refer to our work as graph partitioning active contours (GPAC). Such cost functions have been widely used within the graph partitioning framework but their minimization usually requires certain simplifications, which introduce inaccuracies to the final segmentation. By minimizing pair-wise similarity based cost functions using GPAC, we are able to achieve better segmentation results. Efficient implementations of GPAC are proposed. Finally we show an application in which we use GPAC for pruning categories in large image databases.

Contents

List of Tables **xv**

List of Figures **xvi**

1 Introduction **1**

1.1 Edge-driven Variational Techniques 4
1.2 Region-driven Variational Techniques 6
1.3 Main Contributions 9
1.4 Overview of the Dissertation 12

2 Previous Work on Image Segmentation **14**

2.1 Edge Detection 15
2.2 Variational Image Segmentation and Active Contour Methods . . 18
2.2.1 Edge-based Geometric Active Contour Methods 20
2.2.2 Region-based Geometric Active Contour Methods 23
2.2.3 Anisotropic Diffusion 25
2.2.4 Implementation Issues: Level Set Methods 28
2.3 Image Segmentation using Vector Fields 33
2.3.1 Edgeflow Image Segmentation 34
2.4 Graph Partitioning Methods (GPM) 37
2.5 Other Methods 39

3 Curve Evolution and Anisotropic Diffusion on Natural Images **42**

3.1 Edgeflow-driven Curve Evolution 44
3.2 Comparison of Edgeflow with Geodesic Active Contours 48
3.3 Improving Curve Evolution Stability using Divergence of Edgeflow 52
3.4 Berkeley Segmentation Ground Truth 56
3.5 Evaluation Framework 62

3.5.1 Training Geodesic Active Contours (GAC) 66
3.5.2 Training Gradient Vector Flow (GVF) 69
3.5.3 Training Edgeflow (EF) 72
3.5.4 Test results 75
3.6 Anisotropic Diffusion 79
3.7 Edgeflow-driven Anisotropic Diffusion 82
3.8 Evaluation of Anisotropic Diffusion 83
3.8.1 Training Perona-Malik Flow (PM) 83
3.8.2 Training Self-Snakes (SS) 88
3.8.3 Training Edgeflow 91
3.8.4 Test results 91
3.9 Conclusion and Discussion 93

4 Multi-scale Boundary Detection and Color Image Segmentation 100
4.1 Multi-scale Edgeflow vector field and edge detection 104
4.2 Multi-scale Image Segmentation 112
4.3 Color Image Segmentation 117
4.4 Conclusions 123

5 Graph Partitioning Active Contours (GPAC) 124
5.1 Comparison of ACM and GPM 128
5.2 Curve Evolution based on Minimum Cut Criterion 130
5.3 Maximum Cut and Region Stability 134
5.4 Curve Evolution for Other Pairwise Similarity Based Cost Functions 142
5.4.1 Curve Evolution based on Boundary-normalized Cut . . . 144
5.4.2 Curve Evolution based on Average Cut Criterion 145
5.4.3 Curve Evolution based on Normalized Cut Criterion 146
5.5 Strategies for Integration of Edge Information 146
5.6 Multi-region Segmentation 148
5.7 Efficient Implementation 152
5.8 Multi-scale Strategies 156
5.9 Interactive Image Segmentation 157
5.10 Experimental Results 159
5.11 Discussions 167
5.12 Conclusion 170

6 Search and Category Pruning in Image Databases using Segmentation 173
6.1 Category-based Database Design and Image Retrieval 175

6.1.1 Image Ingestion 176
6.1.2 Database Organization and Image Search 177
6.1.3 Examples of Search and Retrieval 181
6.2 Category Pruning 182
6.2.1 Segmentation Framework 186
6.2.2 Pruning a Category 188
6.2.3 Experimental Results 190
6.2.4 Discussion 198

7 Future Work and Conclusion 200
7.1 Texture Edges 201
7.2 Knowledge Discovery from Image Categories and Segmentation . . 204
7.3 Conclusion 205

A Euler-Lagrange of a Functional 207

B First Variation of a Functional 208

Bibliography 210

List of Tables

3.1 Optimum values for each image at $\sigma = 0.75$ using GAC 68
3.2 Summary of the optimum values for curve evolution 79
3.3 Summary of the optimum values for anisotropic diffusion 92

List of Figures

2.1 Demonstration of Edgeflow vector field. a) An image of blood cells. White rectangle marks the zoom area. b) Edgeflow vectors corresponding to the zoom area on the blood cells image. 34
2.2 a) Difference of offset Gaussians. The distance of the centers is decided by the scale parameter. This function is the weighting factor used in calculating the Error. b) The center area is compared to the half rings 1 and 2 at all angles to find the direction of Edgeflow vectors. 35
2.3 The direction with the highest difference in gray scale is not the direction with highest certainty. The direction from gray level 50 to 100 is the direction with highest certainty. The numbers in this figure corresponds to gray scale intensities. 36

3.1 Illustration of the case where the curve is stuck in the background outside the reach of the vector field. The vector field is not able to pull the curve towards the edges. 46
3.2 a) Initialization of the curve evolution. b) Segmentation result using Edgeflow. c) Segmentation using only the conservative component of Edgeflow. d) Geodesic active contour segmentation . . . 51
3.3 a) Binary image and the zoom rectangle, b) Edgeflow $\vec{S}$, c) conservative component of Edgeflow $S_{con} = \nabla V$, d) solenoidal Component of Edgeflow $S_{sol} = \vec{S} - \nabla V$, e) $\nabla\|\nabla\hat{I}\|$, similar to what is used in geodesic active contours. 51
3.4 a) Original binary image. b-k) Images with random noise added increasingly. 53

3.5 a) R6 segmented using geodesic active contours. b) Canny edge detection on R6. c) R8 segmented using Edgeflow-based curve evolution. d) Canny edge detection on R8. e) R10 segmented using Edgeflow-based curve evolution. f) Canny edge detection on R10. 54
3.6 a) Blood vessel image; b) Divergence indicator function of $-\Delta g$ (white corresponds to low values, black to high values); c) Divergence indicator function of $\nabla\|\nabla\hat{I}\|$ (without nonlinear scaling); d) Divergence indicator function of $\nabla\cdot\vec{S}=\Delta V$; e) Divergence of $\nabla\cdot\vec{S}=\Delta V$; f) Edges detected using the original Edgeflow method (without vector propagation) [1]. 55
3.7 a) An edge profile, b) ∇V 56
3.8 Training set of images and their IDs in Berkeley segmentation data set. 58
3.9 Curve instantiation setup. a) A multi-part curve, b) 4 multi-part curves each with 4 sub-parts, c) 9 curves total: 4 multi-part curves and 5 single-part curves. 66
3.10 Performance of Geodesic Active Contours on the training set. . . 67
3.11 Performance with the change of α for $\sigma = 0.75$ (GAC) 69
3.12 Performance of GVF on the training set. 71
3.13 Performance of EF on the training set. 73
3.14 Performance comparison for $\xi = 0.5, 0.75, 1$ 74
3.15 Performance of EF2 on the training set. 75
3.16 Performance comparison of EF, EF2, GAC and GVF on the training set. 76
3.17 Performance comparison of EF and EF$'$. 77
3.18 Performance comparison of EF2 and EF2$'$. 77
3.19 Performance comparison of EF$'$, EF2$'$, GAC and GVF on the training set. 78
3.20 Curve evolution results on the test set. First column: GAC, second column: GVF, third column: EF$'$ and forth column: EF2$'$ 80
3.21 Self-snakes image diffusion. First two rows correspond to the diffusion and edge function g for the case when β is set to zero. Third and forth rows correspond to the diffusion when α is set to zero. . 84
3.22 Edgeflow-driven image diffusion for $\alpha = 0$. First row shows the diffusion process and the corresponding edge functions V are given in the second row. 85

3.23 Flowchart of the algorithm for capturing thin boundaries from piecewise homogenous diffused image, generated by anisotropic diffusion. 89
3.24 Performance of self-snakes on the training set. 90
3.25 Performance comparison of EFD versus EF2D on the training set. 92
3.26 Anisotropic diffusion results on the test set. First column: PM, second column: SS, third column: EFD and forth column: EF2D 94
3.27 EF2D results on the test set. 95
3.28 EF2D results on the test set. 96

4.1 Demonstration of good localization of multi-scale edge detection. Results show that multi-scale edge detection favors edges that exist at both fine and coarse scales and edges are localized at the finer scale. a) Original image. b) Edge strengths at spatial scale $\sigma = 1$ pixel. Some clutter in the background is recognized as edges. c) Multi-scale edge detection using scales from $\sigma = 1$ to $\sigma = 4$. d) Edge strengths at scale $\sigma = 4$. Edges are not well localized. 101
4.2 Demonstration of multi-scale edge detection. First column shows the original images. Second column shows the edge strengths at spatial scale $\sigma = 1$ pixel. Third column shows multi-scale edge detection results using scales from $\sigma = 1$ to $\sigma = 4$. Fourth column shows edge strengths at scale $\sigma = 4$ pixels. 110
4.3 Demonstration of multi-scale edge detection. First column shows the original images. Second column shows the edge strengths at spatial scale $\sigma = 1$ pixel. Third column shows multi-scale edge detection results using scales from $\sigma = 1$ to $\sigma = 4$. Fourth column shows edge strengths at scale $\sigma = 4$ pixels. 111
4.4 a) Original image. b) Segmentation result at spatial scale $\sigma = 1.5$. c) Segmentation result at spatial scale $\sigma = 6$. d) Multi-scale segmentations using scales $\sigma = 1.5$ to $\sigma = 6$. e) Detail around the head area. f) Detail for segmentation at $\sigma = 1.5$ g) Detail for segmentation at $\sigma = 6$ h) Detail for segmentation using scales 1.5 to 6. 114
4.5 a) Original image. b) Segmentation result at spatial scale $\sigma = 1.5$. c) Segmentation result at spatial scale $\sigma = 6$. d) Multi-scale segmentations using scales $\sigma = 1.5$ to $\sigma = 6$. e) Detail around head area for $\sigma = 6$. f) Detail for multi-scale segmentation. 115

4.6 First column shows the segmentation results at spatial scale $\sigma = 1.5$. Second column shows the corresponding multi-scale segmentations using scales $\sigma = 1.5$ to $\sigma = 6$. 116
4.7 a) Segmentation at spatial scale $\sigma = 1.5$. b) Multi-scale segmentation using scales $\sigma = 1.5$ to $\sigma = 6$. 117
4.8 First column shows the segmentation results using gray scale features at spatial scale $\sigma = 1$. Second column shows the corresponding color segmentations. Third column shows the color anisotropic diffusion result. 121
4.9 First column shows the segmentation results using gray scale features at spatial scale $\sigma = 1$. Second column shows the corresponding color segmentations. Third column shows the color anisotropic diffusion results after convergence. 122

5.1 a) A large balanced curve initialized. b) Foreground object captured correctly. c) Small curve initialized within the foreground. d) Initial curve shrinks and disappears 136
5.2 Demonstration of how noise can effect the curve evolution. a) Single curve initialized overlapping both the foreground and the background. b) Corresponding curve evolution result (all white points correspond to the curve). c) A multipart curve initialized in a grid fashion. d) Corresponding curve evolution result (all white points correspond to the curve). The curve evolution on a noisy image splits the curve into many small pieces. 139
5.3 Three figures on each row, where each row corresponds to maximum cut evolution under 4 different type of curve instantiations. First column corresponds to various initializations of the curve. Second column shows a state of the curve during the evolution. Third column shows the segmentation result at convergence. This example demonstrates the stability of maximum cut framework with respect to curve instantiation and noise. 140
5.4 Segmentation of an image corrupted by Gaussian noise and applied an illumination effect. a) Original image. b) Corresponding curve evolution result using normalized maximum cut segmentation. . . 141
5.5 a) Original image (266x200), b) initial curve consisting of 144 subparts, c) maximum cut segmentation and corresponding, d) foreground, e) background, f) normalized maximum cut segmentation and corresponding, g) foreground, h) background. 143

5.6 a) Original image, b) Segmentation result without edge integration, c)Segmentation result with edge integration. 147
5.7 a) Original image and the user drawn rectangle, b) Segmentation result without user interaction, c)Segmentation result using the user drawn rectangle. 159
5.8 Segmentation of an arterial tree image. a) Original image. b) Curve initialization. c) Curve is evolving. d) Segmented background. e) Segmented foreground. f) Segmentation boundary. 161
5.9 Foreground/Background segmentation. First column shows the original images. Second and third columns correspond to foreground and background. 163
5.10 Foreground/Background segmentation. First column shows the original images. Second and third columns correspond to foreground and background. 164
5.11 Demonstration of improvements using texture feature vectors. a) Original image. b) Color segmentation. c) Texture segmentation using Gabor texture features at 3 scales and 6 orientations. d) Segmentation using texture and color features. Texture features weighted 70% and color features 30%. 165
5.12 Multi-region segmentation using coupled curve evolution a) Original image. b) Segmentation boundaries. c) Background region. d-f) foreground regions each corresponding to the interior of the curve. 166
5.13 Segmentation results for the multi-region approach using coupled curve evolutions. 9 curves are initialized for each image. 167
5.14 Segmentation results for the multi-region approach using coupled curve evolutions. 9 curves are initialized for each image. 168
5.15 Comparison of background/foreground segmentation vs. multi-region segmentation using couple curve evolutions. First column shows the original images. Second and third columns correspond to foreground and background. Forth column shows the multi-region segmentation boundary. 169
5.16 Segmentation of an image with same mean but different variations in its two regions a) Initial curve. b) Segmentation result. 170
5.17 Reproduced from ratio cut paper [2]. a) Original image. b) Ratio cut segmentation after 4 bi-partitioning steps. 170

6.1 Database structure. 178

6.2 Random images from categories identified using the keyword “hat”. The top right image is selected for a constrained content-based search. 183
6.3 Results of a constrained content-based search using the image selected in Fig. 6.2. Top row shows query image and 5 most similar images with respect to the color descriptor. Bottom row shows query image and 5 most similar images with respect to the texture descriptor. The bottom right image is added to the query. 183
6.4 Results of constrained content-based search using the two images selected in Fig. 6.3. Relevance feedback is used to compute the query vector and relative feature weightings. 184
6.5 Results of an unconstrained content-based search using the same query vector and feature weightings as the search in Fig. 6.4. . . . 184
6.6 Initialization of the curve evolution for image segmentation. . . . 187
6.7 First column shows the original image. Foreground is shown in second column. Third column show the signed distance map. . . . 191
6.8 First column shows the original image. Foreground is shown in second column. Third column show the signed distance map. . . . 192
6.9 Average distance map. 193
6.10 First column shows the original image. Initial foreground is shown in second column. Foreground after using spatial relations is shown in third column. 194
6.11 a-d) Images with smallest distances. e-h) Images with the largest distances. 196
6.12 a) Histogram of the distances. b) Precision-Recall curve. c) Precision vs. cutoff point. d) Recall vs. cutoff point. 197
6.13 Images that could be misclassified. 198

7.1 a) Cheetah image. b) Edge function generated from Edgeflow vector field with a Gaussian offset of 4σ using texture features. c) Edge function generated from Edgeflow vector field with a Gaussian offset of σ using texture features. 203

Chapter 1

Introduction

The primary goal of this thesis is to develop robust image segmentation methods based on variational techniques. Image segmentation is one of the fundamental problems in image processing and computer vision. Segmentation is also one of the first steps in many image analysis tasks. Image understanding systems such as face or object recognition often assume that the objects of interests are well segmented. Different visual cues, such as color, texture and motion, help in achieving segmentation. Segmentation is also goal dependent, subjective, and hence ill-posed in a general set up. However, it is desirable to consider generic criteria that can be applied to a large variety of images and can be adapted for specific applications. This thesis work focuses on developing such segmentation methods that work on natural images. Given an image, the goal is then to find a partitioning of the image such that the pixels within the interior of the regions are *similar* to each other and are dissimilar across the region boundaries. While this is not a very technical definition of segmentation, it offers an intuitive explanation of the problem and we hope to make this precise as we consider specific notions of similarity in the following discussions.

Existing methods for image segmentation can be broadly classified as region-based or edge-based. An example for an edge-based technique would be a method that defines the regions in a way such that the boundaries pass through the strongest edges. An example for a region-based technique would be a method that partitions the image into regions such that given the number of regions, the sum of the variances within the regions is minimized.

It has been argued that, region features and edge features contain independent information that can help image segmentation. There are many techniques that create a segmentation by utilizing both edge and region based features or constraints. On the other hand, these formulations may be very difficult to solve and may require additional constraints or assumptions to further simplify the problem. A more common approach is to feed the output of a region-based system to an edge-based system or vice versa. One main problem with combining edge and region based segmentation is that it is difficult to measure the contribution of the edge and region based components to the final segmentation. This becomes an issue when the algorithm needs to be optimized or improved upon later on.

There are low level cues that are used for both edge-based and region-based segmentation methods. A measure or a technique for calculating continuous edge strengths/probabilities (such as image gradient or Earth mover's distance [3] between pixel neighborhoods [4]) is needed for edge-based segmentation techniques. For region-based methods, either a similarity measure for grouping pixels or a region model or a set of models that these methods follow is required.

In the last decade, variational methods for image segmentation has been extensively studied. Variational methods offer a strong mathematical basis for developing image processing algorithms. From the mathematical design point of view, variational techniques can be divided into three groups:

- Diffusion-based techniques.
- Curve evolution techniques.
- Techniques that are based on region models.

Diffusion-based techniques are based on diffusing information from a pixel in the image to its neighbors. This usually results in a smoothing of the image. Curve evolution techniques attempt to evolve a closed contour over the image domain. The evolution of the curves is guided by some attraction forces that might be generated from edge or region cues. There are a few techniques that try to unite these two methods by considering the level sets of the image as closed curves and formulate the diffusion problem as the curve evolution of these level sets. On the other hand, techniques that are based on region models approach the segmentation problem from a generative point of view. The image surface is intended to be regenerated by a combination of edges (lines) and smooth patches from which the segmentation is driven. Depending on the image features that the variational formulation utilizes, each of these techniques can also be classified as either edge-based or region-based.

In this thesis, we investigate the behavior of variational methods on natural images and address their shortcomings by introducing new techniques. One of the main problems with current variational techniques is their limited application and evaluation on natural images that are usually rich in texture. This is especially true for edge-based methods due to the difficulty of capturing correct edges in complex images. As an example; anisotropic (edge preserving) diffusion can smooth and eliminate noise or clutter in the image while sharpening certain strong edges, but it hasn't been empirically investigated if anisotropic diffusion also eliminates some real object boundaries that might be crucial for high level tasks such as object detection. Similar issues apply to edge-based curve evolution techniques. Edge-based curve evolution techniques are extensively used in medical image analysis. One of the shortcoming of edge-based curve evolution is boundary leaking through weak edges and these type of images usually contain objects with clear boundaries. On the other hand, curve evolution on natural images is more likely to leak through missing or weak edges.

1.1 Edge-driven Variational Techniques

In the early 80's, Witkin [5] wrote the first paper on the Gaussian scale space. The Gaussian scale space is generated by continuously filtering the image by Gaussians with increasing variance. This causes the image to be increasingly smoothed. The behavior of the edges or the zero crossings after applying a Laplacian has been investigated by many researchers during the 80's. The main observation was that continuous smoothing eliminates noise and weak edges, but at the same time, the remaining edges are being shifted from their original locations. This results in poor localization of the edges, which is not desirable. This Gaussian scale space representation can be equivalently formulated as a 2D heat diffusion of the image where the time parameter corresponds to the variance of the Gaussians. Perona and Malik [6] extended this formulation to nonlinear heat diffusion where the diffusion is stopped or slowed down at the high gradient locations of the image. This formulation has a better edge localization since the edges are not allowed to diffuse. Later on it has been shown that this formulation is not well-posed and might cause undesirable results such as sharpening the noise if signal to noise ratio is low [7]. One way to fix this is by using Gaussian smoothing before taking the gradient of the image.

One important observation about Perona-Malik formulation is that the diffusion can be decomposed into two orthogonal components, one along the gradient direction and one orthogonal to the gradient. The flow in the direction of the gradient is similar to shock filtering and can be used to sharpen edges whose strength is above a certain threshold. Shock filtering was introduced by Osher and Rudin [8] for deblurring the image. The other term orthogonal to the gradient is used for smoothing along the edges and can be used as an edge preserving smoothing term. The problem with Perona-Malik flow and other anisotropic diffusion formulations is that both diffusion and sharpening are controlled by a single edge strength function. First of all, it is difficult to control the level of smoothing versus sharpening with a single parameter. Saint-Marc et al. [9] observed in their implementation of anisotropic diffusion that while edges are sharpened quickly, it takes a long time for smoothing. Secondly, this formulation introduces second or higher order derivatives of the image to the sharpening term and it is well known that high order derivatives of the image increase the noise sensitivity. In this context, our proposed methods use only first order image derivatives in the sharpening terms in our diffusion equations. Further, we do not use gradient-based edge strength function as the initial condition. Instead, our anisotropic formulation is based on the Edgeflow vector field [1]. This Edgeflow based segmentation has been successfully applied to a large class of natural images.

Similar ideas also apply to the edge-based curve evolution. The partial derivative equations for the smoothing term that is tangential to the level lines of the image can be used to keep the curve smooth during the evolution. Similarly, the same partial differential equations for the edge sharpening term can be used to stop the curve evolution at the edges.

1.2 Region-driven Variational Techniques

Region-based techniques are based on defining a model for the image or defining an energy functional that is minimized by a partitioning of the image. The most influential of these models is proposed by Mumford and Shah [10]. Mumford-Shah functional models the image as a piecewise smooth function where these smooth regions are separated by discontinuities (jumps in image intensities). The objective then is to minimize the length of the discontinuity while maximizing the smoothness of the piecewise functions that are approximating the original image as best as possible. To simplify this formulation, it is common that the approximating functions are taken as piecewise constant. Inspired from this framework, Chan et al. [11] defined a region-based curve evolution framework where interior and exterior of a curve are modeled by constant functions (comes out as the average feature values of these regions). Similar frameworks are also defined for piecewise smooth functions [12, 13]. So far our discussion has been about functionals that are trying to model the interiors of the regions, thus increasing the intra-region similarity. Yezzi et al. [14] defined a segmentation cost function and associated curve evolution that is based on maximizing inter-region dissimilarity. In their work, the curve is evolved in a direction such that the separation of the means of the interior and exterior of the curve is maximized.

Natural images are quite complex and can not always can be well modeled by these cost functions. Even if they can be modeled, finding a solution that minimizes the original Mumford-Shah functional is computationally expensive and could become very difficult. Another very popular image model is using pairwise similarities of pixels in an image. This approach models a region by the pairwise similarities within the region. Intra region similarity is high when the sum of the pairwise similarities within the region is high, and inter-region dissimilarity is maximized when the pairwise similarities across the boundary is low. This type of cost functions haven't been attempted by the variational techniques previously. On the other hand, graph partitioning methods have used such pairwise similarities extensively and successfully. The advantage of such cost functions is that there is no need to make assumptions about the characteristics of the regions. Many different models can be realized by defining the similarity and dissimilarity measures between pixels. These measures can even be based on domain knowledge. The downside is their computational complexity. Finding a minimizer for these cost functions can be computationally intensive and usually require simplifications driven by computational needs. For example, spectral graph partitioning methods define the similarity only within the neighborhood of a pixel, e.g. 20 pixel radius, and similarity to other pixels are set to zero. such approximations often lead to incorrect results such as over-segmentation of large homogenous areas.

We believe that pairwise similarity based cost functions are superior to the previous variational region models and offer better flexibility. This can be confirmed by the fact that majority of the recent segmentation work that has shown promising results on natural images are based on pairwise similarities. We introduce our *Graph Partitioning Active Contours* that uses a cost function defined on pairwise dissimilarities between pixels. We then solve this cost function using variational and curve evolution techniques. Contrary to graph partitioning methods, we are not limiting the similarities to a neighborhood of the pixels. This way we are able to better approximate the correct solution. We refer to this framework as the *Maximum Cut framework*. Another important contribution is that we propose novel, fast and accurate implementation techniques for the minimization of pairwise similarity based cost functions. Unlike graph partitioning methods, our framework is not specific or limited to a single cost function. Most cost functions that are based on pairwise similarities can utilize our framework with minimal changes.

1.3 Main Contributions

The overall contribution of this thesis is the introduction of new techniques for applying variational segmentation to natural images.

- **Curve evolution and anisotropic diffusion using Edgeflow vector field**

  Edgeflow technique [1] has been shown to work successfully on natural images. On the other hand, to generate a segmentation from edge detection, Edgeflow needs to utilize a heuristic edge linking stage. Variational techniques such as curve evolution and anisotropic diffusion produce closed boundaries without any extra steps. Based on these observations we propose a new curve evolution and a new anisotropic diffusion that are based on Edgeflow vector field. We demonstrate by evaluations on Berkeley segmentation data set that our methods outperform the state of the art for segmenting natural images.

- **Multi-scale Edgeflow vector field for edge detection, curve evolution, anisotropic diffusion and image segmentation**

  A good localization of the edges is very important for the quality of image segmentation. This requires detecting the edges at a small spatial scale. However, at small scales most of the clutter and noise are also detected as edges. Usually when scale is increased, these spurious edges quickly disappear. Unfortunately, increasing the scale also shifts the real edges thus resulting poor localization. To solve these issues, we propose a multi scale extension of the Edgeflow method. The original Edgeflow method is defined only for a single scale parameter. When the scale is increased, the Edgeflow method suffers from pour localization while small scales introduce unwanted detail. Our multi-scale approach offers excellent localization of the edges while eliminating the clutter. Consistency among vector fields generated at multiple scales are analyzed and edges that exist at both coarse and fine scales are favored. These edges are then localized at the finer scales. The second advantage of this multi-scale approach is that the reach (coverage area) of our vector field is much larger than Edgeflow vector field that is generated at a small scale. For the tasks of curve evolution and anisotropic diffusion, the reach of the vector field is as important as good localization of the edges. For example, in the case of curve evolution if the reach of the vector field is larger, curves can be pulled to the boundaries from greater distances and can be forced to stay at the boundary much more easily.

- **Graph partitioning active contours: variational solution to pairwise similarity based cost functions**

  One of the problems in region-based variational methods is that the region models and segmentation cost functions, despite being theoretically very nice, have not been widely and successfully applied to natural images. On the other hand, pairwise similarity based cost functions have become increasingly popular in the last few years and have been applied to a wide variety of natural images successfully. In this thesis we provide solutions to such complex cost functions within the variational framework. The variational solutions of such functions also makes certain global cost functions and their solutions possible, which are not practical within the spectral graph partitioning framework.

- **Empirical evaluation of variational segmentation methods**

  One the shortcomings of variational segmentation methods is that they are not widely applied to natural images. We are not aware of any extensive evaluation of various anisotropic diffusion and curve evolution techniques for the segmentation of natural images. Most evaluation of variational techniques has been done on synthetic images with added noise or on certain medical images. Typically, results of edge detection or image smoothing are presented, making an objective evaluation difficult. As we demonstrate, the usual assumptions do not easily generalize to natural images. To address these issues, we propose a new evaluation methodology for variational segmentation. We empirically evaluate both edge-based curve evolution and various anisotropic diffusion techniques by conducting extensive tests within this framework.

- **Application: Category pruning in image databases**

  A practical application of image segmentation to unsupervised pruning of categories in an image database is presented. A category in an image database is defined as a set of images that are all associated with a concept. A category for example can be a directory on a user's computer that is full of images. Another example is a Google image search for a keyword that will return a set of images that has this keyword associated with them. It is also possible to populate an image database and categorize the images at the same time. We collected more than *600,000* images from internet automatically and categorized using an existing categorization of *urls*. Automated categorization is usually not perfect as some post processing and pruning needed to increase the quality of these categories. Despite being a difficult problem, this is essential for content based image retrieval and browsing of images. To achieve this goal, we first apply a foreground/background segmentation to all images in a category. Based on this segmentation boundary, each image is associated a signed distance function that characterizes the spatial relation between the foreground and the background. Averaging individual signed distance maps, a distance map for the category is defined. Utilizing this signature distance map of the category, the images are re-segmented. We show that we are able to both increase the quality of the segmentations and increase the precision of a category with minimal effect to the recall.

1.4 Overview of the Dissertation

An overview of the rest of this dissertation is as follows. Chapter 2 provides a survey of approaches that have been proposed for image segmentation. Edge detection, curve evolution, variational segmentation, and recent successful image segmentation techniques are discussed. In Chapter 3, we set up an evaluation framework for variational image segmentation schemes. We focus on two of the main problems, namely edge-based curve evolution and anisotropic diffusion. The objective of this evaluation is twofold: first, we would like to measure the relative performance of various methods, secondly and more importantly, we analyze the performance behavior of these techniques when a certain parameter such as the smoothing scale, sharpening level or gradient threshold is slowly changed. Based on the insight we gather, we propose curve evolution and anisotropic diffusion techniques that incorporate the successful Edgeflow vector field. Chapter 4 presents a multi-scale approach for extending Edgeflow such that the localization of the edges are preserved while the unwanted clutter is removed. Results on natural images using color is also presented. Chapter 5 describes a region-based curve evolution method named Graph Partitioning Active Contours. Unlike previous methods, which attempt to model the image regions with piecewise constant or Gaussian models, a new four dimensional energy functional that is based on a similarity metric between pixels is introduced and its steepest descent minimization is shown. Chapter 6 describes an application to a large web-based image database in which images are automatically categorized into semantic categories. Segmentation is used to identify and eliminate outliers in these categories, thus increasing the precision of the category with little effect to recall. Finally, Chapter 7 concludes with future directions.

Chapter 2

Previous Work on Image Segmentation

Image segmentation has a rich history in image processing research. Many different techniques that work on various image features such as brightness, color, and texture, have been proposed. One aspect that makes image segmentation ill-posed and difficult is its domain and application dependence. For natural images, a combination of gray scale, color and texture is shown to work well. For aerial or remote sensed imagery, texture features show a good performance. For other domains, such as medical images, or images of man made objects, gray scale intensities are mostly sufficient. In this chapter we will review some of the well known image segmentation methods, with emphasis on variational techniques.

2.1 Edge Detection

Early work on segmentation can be traced back to edge detection research in the 1970s. The resultant edges are typically disconnected and hence do not usually lead to proper image segmentation. Ad hoc techniques are often used for tracing and connecting the edges. Research on edge detection dates back to early image processing in the 1970s. Early methods are based on accurately estimating the gradient maxima of an image. Marr and Hildreth [15] used zero crossings of the Laplacian of the Gaussian (LOG) filtered output as the edges. Later on in the 80's this technique raised more interest for the multi-scale analysis of images. Even though closed contours are detected, due to the use of the second derivative, LOG is not robust in the presence of noise in images. Canny [16] showed that LOG has poor signal to noise ratio and poor localization. He derived an optimal linear edge detection filter for detection of step edges corrupted by white noise. Optimality is defined in terms of good localization of edges, high detection rate, and single detection. The resulting filter is very similar to the first derivative of a Gaussian and this is the commonly used form of this edge detector. Canny's edge detector is still widely used and almost every edge detection paper compares the results to the Canny's. The main idea in all these techniques is to create a scalar function, which defines the edge strength (and orientation in some cases) at each pixel (edgeness of a pixel). Edges are then found by non-maxima suppression and (hysteresis) thresholding of the edge function.

Another noteworthy mention is Haralick's method [17] of locally fitting a linear combination of discrete bases of Chebychev polynomials and calculating the first and second directional derivatives to locate edges. Canny shows [16, Chapter 7] that the optimal detector for Haralick's work that is attained for up to third order polynomials is very similar to the first derivative of Gaussian and Canny's optimal detector.

Canny's work was based on designing an optimal linear filter for step edges. Since then, many attempts has been made to create non-linear filters that perform better than linear filtering. One such approach is using quadrature pairs of even and odd symmetric filters [18, 19]. Perona and Malik [19] show that using second derivative of Gaussian oriented at several directions as even symmetric filters, and their Hilbert transforms as odd symmetric filters, one can detect a combination of lines and step edges. Thi is in contrast with detecting only the step edges using Canny's detector, and at the same time the filters do not output a response for areas with constant gradient (such as those caused by illumination).

Another nonlinear edge detector is the SUSAN approach [20]. In this work, the center point of a circular area is compared to the other pixels in the circle for matches. The value of total matches is minimized at an edge point and further minimized at a corner point. The advantage of this approach is the lack of derivative operations, which helps reduce the noise sensitivity. Another interesting nonlinear detector is the Nitzberg edge detector [21]. This detector is based on calculating the 2×2 matrix $N(x, \sigma) = G_\sigma * (\nabla I \cdot \nabla I^T)$. The eigenvalues of this matrix give the maximum and minimum rate of change. This type of detectors have been widely used for corner detection [22]. Nitzberg's edge detector is able to achieve good texture discrimination. More recently, nonlinear edge detectors that are based on histogram or distribution distances are proposed [23, 24] and mostly shown to work well on color and texture features. In a circular neighborhood, half of the circle is compared to the other half for changes in feature distributions. The common distance measures are earth mover's distance [3] or χ^2 distance. Other non-linear edge detectors include many anisotropic diffusion techniques that are discussed in Section 2.2.3.

Many papers are devoted to the evaluation of edge detector performance. An interesting work from Bowyer et al. [25] used receiver-operating-characteristic (ROC) curves to evaluate the relative performance of 8 linear and non-linear edge detectors including Canny's. ROC curves plot percent of false positive edges vs. percent of unmatched ground truth edges. The nonlinear detectors that are evaluated include the SUSAN edge detector and the edge detector based on robust anisotropic filtering from Black et al. [26]. The results show that Canny edge detector performs better than any other edge detector with Heitger edge detector [27] coming very close. See references within this work for older papers on edge detector evaluation. Another work in this area is done by Konishi et al. [28]. This work compares linear and nonlinear operators used in Nitzberg edge detector, Canny edge detector and LOG edge detector in terms of their usefulness for the edge detection task. The findings show that the nonlinear operator used in Nitzberg edge detector performs better for the task of edge detection than the optimal linear operator (derivative of Gaussian) used in Canny's, and both detectors perform significantly better than LOG.

In a recent work, Martin et al. [24] created a comprehensive segmentation ground truth data set [29] and made it available on their web site. The difference of this ground truth data compared to the data sets used in previous evaluation papers is that the images are selected as diverse natural images. A large number of these images are segmented for ground truth, and each image is segmented at least by five different people who are unfamiliar with segmentation research. The authors compared an edge detector that is based on χ^2 distance of distributions within half disks of a circular neighborhoods to Canny's and Nitzberg edge detectors. Improvements are shown in gray scale but more drastic improvements originated from the use of texture and gray scale together, and more improvements are achieved by utilizing a combination of gray scale, color and texture features.

2.2 Variational Image Segmentation and Active Contour Methods

Variational segmentation methods are based on minimizing segmentation cost functions using gradient descent. The segmentation functional might not be given explicitly. In that case, the descent equations are designed using geometric considerations and characteristics of the flow. The three influential papers on variational segmentation are 1) Mumford and Shah's proposal [10] of the well known segmentation functional that is based on modeling the image with piecewise smooth functions; 2) Snakes approach proposed by [30], which attempts to evolve a curve and fit it to the nearby edges; and 3) Anisotropic diffusion from Perona and Malik

[6] that modifies the classical Gaussian scale space approach so that the edges are preserved while homogenous areas are smoothed. An interesting work from Shah [31] attempts to unify these three methods. In this work, level sets of the image are considered as curves that are evolving driven by forces from a segmentation cost function, which in turn causes anisotropic diffusion of the image.

Active contours and curve evolution methods usually define an initial contour and deform it towards the object boundary. The problem is formulated using partial differential equations (PDE). The previous research follows two different paths in terms of representation and implementation of active contours, namely parametric active contours and geometric active contours. Parametric active contours use a parametric representation of the curves, and geometric active contours utilize level set methods [32, 33] for its implementation. The advantage of geometric active contours is that using level set methods, sharp corners and splitting or merging of the curves during evolution are automatically taken care of whereas parametric methods have difficulty handling these situations. Recently some connections between these two methods have been established [34, 35]. While being popular in the early 90's, parametric active contours are being replaced with geometric active contours. The primary reason is that the level set methods naturally handle topology changes in the curve whereas there are implementation difficulties for such cases using parametric active contours. Our discussion in this chapter will be more on the geometric active contours. Our implementation for the methods proposed in this thesis is also based on the level set methods. A summary and comparison of both geometric and parametric active contours can be found in [35].

Kass et al. proposed *snakes* [30], the first curve evolution method, which is designed to fit an elastic curve to the nearby high gradient pixels given an initialization of this curve. At about the same time, Osher and Sethian introduced level set methods [32], which enabled efficient and robust implementation of geometric curve evolution. Geodesic active contours, equivalent of snakes approach for level set method, was proposed in [34, 36]. Compared to non-maxima suppression and thresholding based methods, curve evolution based methods create boundaries with small or no gaps between edges. The downside of this is that if part of the boundary has weak edges, the curve might leak through this opening and this will lead to large errors in boundary detection.

2.2.1 Edge-based Geometric Active Contour Methods

Edge-based active contours aim to identify an object in an image by utilizing the local discontinuities of the image. The idea is to try to fit a closed curve to an edge function generated from the original image. This initial curve is evolved in the direction that minimizes a segmentation cost function. Most of these methods require the curve to be initialized close to the real object boundary. Let $C(\phi) : [0, 1] \rightarrow \mathbb{R}^2$ be a parametrization of a 2-D closed curve. A fairly general geometric curve evolution can be written as:

$$\frac{\partial C}{\partial t} = (T + \beta\kappa)\vec{N} + (\vec{S} \cdot \vec{N})\vec{N} \tag{2.1}$$

where κ is the curvature of the curve, $\vec{N}$ is the *inwards* normal vector to the curve, β is a constant, T is a function defined on the image grid and $\vec{S}$ is an underlying velocity field whose direction and strength depend on the time and position but not on the curve front itself. This equation will evolve the curve in the normal direction. The first term is a speed parameter that expands or shrinks the curve, second term is the curvature flow (geometric heat flow) that makes sure that the curve stays smooth at all times and avoids noisy perturbations, and the third term guides the curve according to an independent velocity field.

In their independent and parallel works, Caselles et al. [37] and Malladi et al. [38] are among the first to use level set methods to segment objects from an image. They initialize a small curve inside one of the object regions and let the curve evolve until it reaches the object boundary. The evolution of the curve is controlled by the local gradient. This can be formulated by modifying (2.1) as:

$$\frac{\partial C}{\partial t} = g(F + \epsilon\kappa)\vec{N} \tag{2.2}$$

where F, ϵ are constants, and $g = 1/(1 + |\nabla \hat{I}|)$. $\hat{I}$ is the Gaussian smoothed image. If F is positive, the curve expands and if F is negative the curve shrinks. This is a pure geometric approach and the edge function, g, is the only connection to the image. The problem with this setup is that if the curve propagates beyond the desired boundary, there is no mechanism to attract the curve back to the desired boundary.

Caselles et al. [34] introduced geodesic active contours, which is an improvement over the previous active contour methods. Starting with the snakes problem defined by Kass et al. [30], they reformulated the energy functional as a minimization of curve length weighted by the edge function g:

$$\underset{C}{Min} \int_0^1 g(C)|C'(\phi)|d\phi \tag{2.3}$$

where $g(C)$ corresponds to g evaluated on the curve C. Minimizing this function via steepest-decent method is equivalent to evolving the initial curve according to:

$$\frac{\partial C}{\partial t} = g\kappa\vec{N} - (\nabla g \cdot \vec{N})\vec{N} \tag{2.4}$$

This approach works similar to the snakes method, but handles topology changes more easily. It requires the initial curve to be close to the boundaries as in the case of snakes because the energy function is minimized locally. Otherwise the convergence of the curve to the object boundary will be too slow, or it might converge to other region boundaries, or more likely get stuck in a homogenous area of the image where there are no edges, hence ∇g is 0. In most curve evolution methods, to solve the slow convergence problem and problems with curve initialization a heuristic constant force term called balloon force [39] is added. This constant speed term is similar to the one from pure geometric methods 2.2. The geodesic active contour equation becomes:

$$\frac{\partial C}{\partial t} = g(F + \kappa)\vec{N} - (\nabla g \cdot \vec{N})\vec{N} \tag{2.5}$$

Note the extra term $-(\nabla g \cdot \vec{N})\vec{N}$ in the curve evolution equation compared to (2.2). $-\nabla g$ defines a vector field on the pixels of the image. The corresponding vectors point towards the closest edge locations. This method is more stable compared to the pure geometric approach.

2.2.2 Region-based Geometric Active Contour Methods

Region-based ACM attempt to partition the image into two regions: foreground and background. They start with an initial closed contour and modify the curve according to the statistics of the interior and exterior of this contour. The curve evolution converges when both these regions become homogenous. One of the advantages of region-based methods is that they utilize the integrity of image features over the whole region as opposed to the local features used in edge-based methods.

Developments in region-based active contours [40] are more recent than their edge-based counterparts. Region-based active contours are less dependent on the initial location of the contour since they don't rely much on the local image features. Also not needing to use the gradient of the image simplifies both the variational formulation and its solution. Region-based methods are also easier to extend to vector valued images such as color and texture images.

Let C be a 2-D closed curve, I be a function defined on the image domain (an open set) R, R_i and R_o be the interior and the exterior of C, m_i and m_o be the corresponding means, A_i and A_o the areas of R_i and R_o respectively, I_i be $I(R_i)$ defined on R_i and I_o be $I(R_o)$ defined on R_o. Yezzi, et al. [14] define their optimization criteria as maximizing the separation of the mean values: $(m_i - m_o)^2$. This leads to a curve evolution equation

$$\frac{\partial C}{\partial t} = (m_o - m_i)\left(\frac{I - m_i}{A_i} + \frac{I - m_o}{A_o}\right)\vec{N} + \gamma\kappa\vec{N} \tag{2.6}$$

where $\vec{N}$ is the normal vector to C, κ is the curvature and γ is a constant weighting factor. The second term is added to keep the curve smooth at all times.

Chan, et al. [11] on the other hand used a limiting version of Mumford-Shah functional [10], where the image is modeled as a piecewise constant function. The general Mumford-Shah functional given below models the image as a piecewise smooth function.

$$\begin{aligned} E(u,K) = \alpha \int_{R\setminus K} \left|\nabla u(x)\right|^2 dx \\ +\beta \int_{R\setminus K} (u-I)^2 dx + \gamma \cdot length(K) \end{aligned} \tag{2.7}$$

where I is the original image, u is the piecewise smooth version of the image, K is the set of discontinuities. The first term forces u to be smooth outside the edges, the second term forces u to approximate I, and the third term forces that the discontinuity set K has minimal length.

Let the discontinuity set K in (2.7) represent a curve C, then minimizing (2.7) is equivalent to finding u that smoothly approximates I_i and I_o. As a limiting case, Chan, et al. [11] uses a modified version of Mumford-Shah functional, where $\alpha \to \infty$. This forces u to be a piecewise constant function. By solving the general Mumford-Shah problem instead of the limiting case and using the curve evolution setup, where the image consists of a foreground and a background, Tsai et al. [12] found the following curve evolution equation

$$\begin{aligned} \frac{\partial C}{\partial t} = \frac{\alpha}{2} \left(\nabla u_i\right|^2 - \left|\nabla u_o\right|^2 |) \vec{N} \\ +\beta \left((I-u_i)^2 - (I-u_o)^2 \right) \vec{N} + \gamma \kappa \vec{N} \end{aligned} \tag{2.8}$$

where u_i and u_o are values of u on the domains R_i and R_o respectively. This requires calculating the smooth estimates u_i and u_o of I_i and I_o at each iteration. This is computationally intensive compared to the method from [11], where I_i and I_o are modeled as constants.

There have been also attempts at combining region-based and edge-based active contours to achieve an hybrid algorithm. Paragios et al. [41] combined geodesic active contours, which is an edge-based method, with the region statistics that are extracted from the image in an off-line fashion. In their work, using minimum description length (MDL) principle, number of regions and statistics for each region are estimated from the image histogram. These statistics are then used in curve evolution step. Using the estimated statistics of each region, independent curves are propagated to extract the boundaries of each region. An example of a curve evolution equation that is tuned to a specific region with estimated statistics is

$$\begin{aligned}\frac{\partial C}{\partial t} &= \alpha \underbrace{\left[G(p(I), \sigma_R) - G(p_k(I), \sigma_R)\right]}_{\text{Region - based force}} \vec{N} \\ &+ (1-\alpha) \underbrace{\left[G(p_B, \sigma_B) \cdot \kappa + \nabla G(p_B, \sigma_B) \cdot \vec{N}\right]}_{\text{Edge - based force}} \vec{N}\end{aligned} \tag{2.9}$$

where G is the gaussian, p is the probability of a pixel belonging to the region of interest, p_k is the probably of a pixel belonging to a region other than the wanted region, p_B is the probability of a pixel being a boundary point.

2.2.3 Anisotropic Diffusion

In the 80's many researchers worked on analyzing the scale space representation introduced by Witkin [5] and the behavior of edges when scale is changed. This included applying the image Gaussian filtering at increasing scales and finding and tracking edges across scales. This Gaussian scale space can be formulated as the evolution of the image under two dimensional heat diffusion

$$I_t = \nabla^2 I$$

where time is analogous to the scale in Gaussian smoothing and initial condition is taken as the image itself. This PDE is equivalent to the flow equation that is the gradient descent of $E(I) = \iint |\nabla I| \, dxdy$.

Perona and Malik [6] modified this flow equation so that image is diffused selectively as opposed to isotropic heat diffusion where information is diffused from a pixel to all its neighbors with equal speed. The modified PDE is:

$$I_t = div(g(\|\nabla I\|)\nabla I) \tag{2.10}$$

Here, g is a stopping function such that $g \to 1$ when $\|\nabla I\| \to 0$ and $g \to 0$ with increasing $\|\nabla I\|$. The selection of g can play an important role on the behavior of anisotropic diffusion. Perona and Malik proposed two different choices for g:

$$g(\|\nabla I\|) = \frac{1}{1 + (\|\nabla I\|/K)^2)} \tag{2.11}$$

and

$$g(\|\nabla I\|) = e^{-(\|\nabla I\|/K)^2} \tag{2.12}$$

K works like a threshold such that strong edges are enhanced and weak edges are smoothed out.

One problem with anisotropic diffusion is the sensitivity to noise. While it works extremely well on simpler images, on natural images or on noisy images it tends to enhance and sharpen noise (especially if signal to noise ratio is low) leading to single pixel sized noise regions at the end segmentation.

Another critique for this anisotropic diffusion is the ill-posedness of the process, meaning two similar images can converge to two completely different results. A discussion about well-posedness of Perona-malik flow can be found in [42, Chapter 3]. A proposal for well-posedness is given by Catte et al. [7], which suggests for the calculation of g, smoothing the image at a fixed scale before taking the gradient (separate from time parameter). Whitaker et al. [43] proposed reducing this scale parameter as t increases.

Saint-Marc et al. [9] introduced an efficient implementation of anisotropic diffusion using recursive filtering. Using Di Zenzo's result [44] for the definition of gradient on vector valued images, Perona-Malik flow was later on applied to color by Sapiro et al. [45] and Gabor texture features by Rubner et al. [46].

A new anisotropic (selective) diffusion scheme, named mean curvature flow, is proposed by Alvarez et al.[47]. This method is based on evolving the level sets of an image with the speed of mean curvature weighted by the stopping term g:

$$I_t = g(\|\nabla G_\sigma * I\|)\|\nabla I\| div \left(\frac{\nabla I}{\|\nabla I\|} \right) = g\kappa \|\nabla I\| \tag{2.13}$$

where κ is the curvature of the level sets.

Osher and Rudin introduced shock filtering [8] for deblurring images. The objective is to sharpen the edges by creating a flow towards the edges at both sides of the edges. The flow equation is given by,

$$I_t = -\|\nabla I\| F(L(I)) \tag{2.14}$$

where $L(I)$ is taken as the second directional derivative [17] and F is chosen such that $sF(s) \geq 0$ for any s.

It has been shown and analyzed by Alvarez et al. [48] and Yu-Li et al. [49] that Perona-Malik flow can be decomposed into two components, one tangent to the level sets of the image (similar to mean curvature flow), and one perpendicular to the level sets (similar to shock filtering). A better way would be creating a flow with again two components, one tangent and one perpendicular, and custom designing each of these components such that they fit the problem at hand (Sharpening the edges while smoothing the homogenous areas). One such flow is self-snakes [50, 51]. This flow is an extension of geodesic active contours such that each level set of the image is evolved with this curve evolution. The resultant flow equations are:

$$I_t = g(\|\nabla G_\sigma * I\|)\|\nabla I\| div \left(\frac{\nabla I}{\|\nabla I\|} \right) + \nabla g \cdot \nabla I \tag{2.15}$$

2.2.4 Implementation Issues: Level Set Methods

In this section we will investigate the challenges associated with the implementation of certain variational problems. Curve evolution is one of these challenging problems and until recently the implementation has been quite tricky and heuristic. If a diffusion framework is based on considering the image level sets as evolving curves, similar issues apply to it. This section is self contained for the implementation of level set methods. For the theory and derivation of these equations, refer to [33].

Within image segmentation, curve evolution problem has been initially proposed using a parametric representation of the curve. This framework is difficult to implement and was not able to handle topology changes without explicit topology tracking. By topology changes we mean splitting a curve into two curves or merging two curves during evolution. These splitting and merging of curves are very common in curve evolution. Initial snakes approach used high order rigidity terms which kept the curve from splitting. Evolving interfaces is a very general problem and is not specific to image processing applications. Another difficulty with curve evolution is the possibility of forming singularities and sharp corners. These are called shocks. Due to the hyperbolic nature of the equations, for an evolution with constant speed in the normal direction of the curve these singularities occur frequently and rapidly. Numerical implementations such as standard central difference approximations are not able to estimate and approximate these situations, build ripples on the curve, and become unstable. To address these problems, Osher and Sethian introduced Level Set Methods [32] in 1988. Later on this technique is also applied to image segmentation by Caselles et al. [37] and Malladi et al. [38].

The idea behind level set methods is: instead of evolving a curve in a two dimensional plane, which requires parametrization of the curve, evolve a 2D function (surface) in 3D. This is a much easier problem. The zero level set of this function $C = \{(x, y) : U(x, y) = 0\}$ smartly arranged to correspond to the curve that we are interested in evolving. Then, evolving $U(x, y, t)$ automatically gives the evolution of C. The evolution of C can be tracked by following the evolution of the zero level set of U. For example, the equivalent level set equation for $C_t = f(x, y)\vec{N}$ can be derived as follows. Suppose $L(t)$ is the evolving zero level set of U, meaning $U(L(t), t) = 0$. Taking the derivative with respect to

t and applying chain rule gives $\nabla U \cdot L_t + U_t = 0$. Taking $C_t = L_t$, we reach $U_t = f(x, y)\|U\|$. We used the relation $\nabla U/\|U\| = -\vec{N}$.

Now, consider a curve evolution equation that is driven by 1) a curvature-based speed term; 2) a constant speed term, where the speed can be a function of the location (x, y). An example is the edge stopping function $g(x, y)$; 3) a term that is based on an advection vector field. This is a very generic curve evolution and can be adapted for use in most segmentation applications. The level set PDE for this curve evolution is:

$$\frac{\partial U}{\partial t} = \alpha\kappa|\nabla U| + \beta F(x, y)|\nabla U| - \gamma\vec{S} \cdot \vec{\nabla} U \tag{2.16}$$

where α, β, γ are constants. F is a constant speed term, and $\vec{S}$ is an external velocity field. Now consider the implementation of this PDE within an $N \times M$ grid (probably the image domain) which contains the curve. First we need to create the function $U(x, y)$ from the curve C. A popular approach is to use the signed distance function. In this case, the value of U at point (x, y) is selected as the minimum distance of (x, y) from C. The sign is selected as positive if (x, y) is outside the curve and selected as negative if the point is inside the curve. After generating U, the evolution is implemented as follows:

$$U_{i,j}^{n+1} = U_{i,j}^{n} + \triangle t(\alpha T_1 - \beta T_2 - \gamma T_3)$$

where i, j are the grid locations and n corresponds to the discretization of time. Let $\vec{S} = (u, v)$. The individual terms are calculated as:

$$T_1 = K_{i,j}^{n}[(D_{i,j}^{0x})^2 + D_{i,j}^{0y})^2]^{1/2}$$

where
$$K_{i,j}^n = \frac{D_{i,j}^{0xx}(D_{i,j}^{0y})^2 - 2D_{i,j}^{0y}D_{i,j}^{0x}D_{i,j}^{0xy} + D_{i,j}^{0yy}(D_{i,j}^{0x})^2}{((D_{i,j}^{0x})^2 + (D_{i,j}^{0y})^2)^{3/2}}$$
(Note that when $K_{i,j}^n$ is inserted, T_1 can be further simplified.) This implementation uses central difference approximations, which can be calculated using:

$$D_{i,j}^{0x} = (U_{i+1,j} - U_{i-1,j})/2$$

$$D_{i,j}^{0y} = (U_{i,j+1} - U_{i,j-1})/2$$

$$D_{i,j}^{0xx} = U_{i+1,j} - 2U_{i,j} + U_{i-1,j}$$

$$D_{i,j}^{0yy} = U_{i,j+1} - 2U_{i,j} + U_{i,j-1}$$

$$D_{i,j}^{0xy} = (U_{i+1,j+1} - U_{i+1,j-1} - U_{i-1,j+1} + U_{i-1,j-1})/4$$

When calculating T_1, it is a good idea to check if $(D_{i,j}^{0x})^2 + (D_{i,j}^{0y})^2$ is less than a very small number such as 10^{-10} and set T_1 to zero if this is so.

$$T_2 = max(F_{i,j}, 0)\nabla^+ + min(F_{i,j}, 0)\nabla^-$$

where

$$\nabla^+ = [max(D_{i,j}^{-x}, 0)^2 + min(D_{i,j}^{+x}, 0)^2 + max(D_{i,j}^{-y}, 0)^2 + min(D_{i,j}^{+y}, 0)^2]^{1/2}$$

$$\nabla^- = [max(D_{i,j}^{+x}, 0)^2 + min(D_{i,j}^{-x}, 0)^2 + max(D_{i,j}^{+y}, 0)^2 + min(D_{i,j}^{-y}, 0)^2]^{1/2}$$

$$D_{i,j}^{+x} = U_{i+1,j} - U_{i,j}$$

$$D_{i,j}^{-x} = U_{i,j} - U_{i-1,j}$$

$$D_{i,j}^{+y} = U_{i,j+1} - U_{i,j}$$

$$D_{i,j}^{-y} = U_{i,j} - U_{i,j-1}$$

and

$$T_3 = max(u^n_{i,j}, 0)D^{-x}_{i,j} + min(u^n_{i,j}, 0)D^{+x}_{i,j} + max(v^n_{i,j}, 0)D^{-y}_{i,j} + min(v^n_{i,j}, 0)D^{+y}_{i,j}$$

At the boundaries of the domain, Neumann boundary conditions are used, which means the grid needs to be extended by 1 in all sides and the value at the boundary is repeated. This extra boundary of width 1 needs to be updated accordingly after each iteration. The size of the time step is calculated following the Courant-Friedrich-Lvy (CFL) condition $\triangle t \leq 1/max(\alpha T_1 - \beta T_2 - \gamma T_3)$. The maximum is taken over all grid points. The curve should not move more than one grid point.

Evaluating U at all grid points is an overkill for curve evolution. A better approach is the narrow band level set methods, which selects the domain as a narrow band around the curve. A band size of 2 to 6 pixels (we assumed $\Delta x = 1$) at both sides of the curve is reasonable. A land mine area of size 1 or 2 is also needed outside the narrow band to check if the curve left the narrow band. If the curve reaches the land mine area, the narrow band need to be re-initialized. The narrow band and the land mine area together build the grid where U is evaluated. As in the general case, the boundary conditions need to be satisfied by repeating the value at the edges after each iteration. Besides the computational advantage of evaluating U on a smaller grid, when narrow band method is used, the time step is dynamically calculated using the narrow band as opposed to full grid and this potentially speeds up the curve evolution.

One price we pay with the narrow band method is the cost of constant reinitialization of the narrow band using signed distance function. It might be a good idea to use Fast Marching Methods [33, Chapter 8] for calculating the signed distance function. More recently, using Additive Operator Splitting (AOS) scheme, potentially faster implementations of narrow band methods are also proposed [52].

2.3 Image Segmentation using Vector Fields

Flow-field based segmentation methods are also proposed within the last decade. Two of the better performing ones are the Edgeflow [1] and the Tabb and Ahuja's method [53]. Especially Edgeflow segmentation has been shown to work well on thousands of real life images by utilizing color and Gabor texture features. Both vector fields are generated directly from the image and they have similar characteristics. Edgeflow vector field is generated as follows: an estimate for the best direction to the closest boundary at a specified spatial scale is selected as direction of the vectors. Then the directional gradient of the smoothed image is used as the vector magnitudes. In Tabb and Ahuja's method, for each pixel, the pixel's feature difference to other pixels in its local neighborhood is calculated and a vectorial sum to these pixels weighted by the feature difference gives the vector field.

There are also several other vector fields that are utilized in curve evolution. These vector fields are generated from the edges in the image. We mentioned $-\nabla g$ in the context of geodesic active contours. This vector field is generated from the image gradients. Cohen and Cohen [54] first apply a Canny edge detector [55] to the image to generate an edge indicator function. From these edges, a scalar potential function is computed using a distance map. The negative gradient of this function is used as the vector field. Gradient vector flow [56] creates its vector field

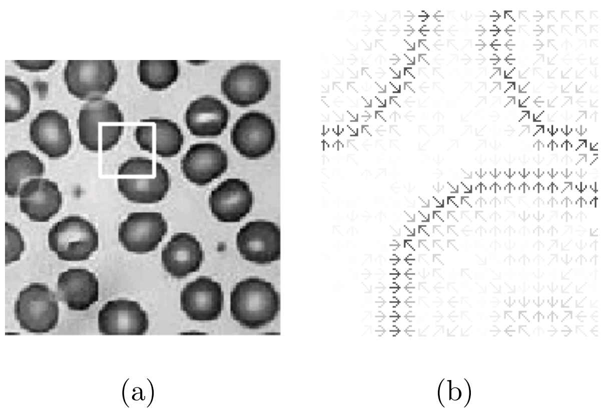


Figure 2.1: Demonstration of Edgeflow vector field. a) An image of blood cells. White rectangle marks the zoom area. b) Edgeflow vectors corresponding to the zoom area on the blood cells image.

by minimizing an energy functional on an edge function. The vectors resemble $-\nabla g$ around the edges. At the homogenous areas away from the edges, the vectors are interpolated from the edges through the solution of a Laplace equation. Both Cohen's vector field and the gradient vector flow cover the full image, meaning from any point on the image, an edge pixel can be reached by following the vector directions. All these methods are computed from a predefined edge function, so they cannot bring more information about the image that is not contained in this edge function.

2.3.1 Edgeflow Image Segmentation

The defining characteristic of Edgeflow is that the direction of the vectors point towards the closest edges at a predefined spatial scale. An example of Edgeflow vector field is shown in Fig. 2.1. Assume that $s = 4\sigma$ is the spatial scale at which we are looking for edges. Let $\hat{I}_\sigma$ the smoothed image with a Gaussian of variance

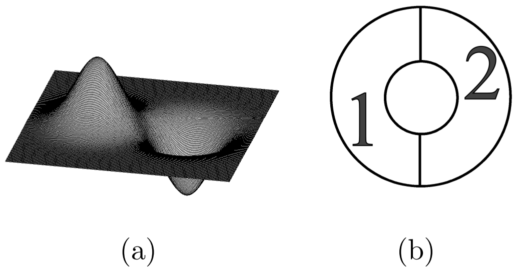


Figure 2.2: a) Difference of offset Gaussians. The distance of the centers is decided by the scale parameter. This function is the weighting factor used in calculating the Error. b) The center area is compared to the half rings 1 and 2 at all angles to find the direction of Edgeflow vectors.

σ^2. At pixel $p(x, y)$ and orientation θ, the prediction error is defined as:

$$Error(\sigma, \theta) = \left| \hat{I}_\sigma(x + 4\sigma\cos\theta, y + 4\sigma\sin\theta) - \hat{I}_\sigma(x, y) \right| \tag{2.17}$$

This can be seen as a weighted averaged difference between the neighborhoods of two pixels and the weighting function is shown in Fig. 2.2(a). Instead of finding the direction with the largest error value, we define an edge certainty in the direction θ using relative errors in the opposite directions θ and $\theta + \pi$:

$$P(\sigma, \theta) = \frac{Error(\sigma, \theta)}{Error(\sigma, \theta) + Error(\sigma, \theta + \pi)} \tag{2.18}$$

Fig. 2.3 demonstrates how using relative certainty is different from the direction of highest change. In this figure the numbers correspond to gray scale intensities.

Using these certainties at each direction, the probable edge direction at a point $p(x, y)$ is estimated by analyzing the total certainties over half circles:

$$\arg\max_{\theta} \int_{\theta-\pi/2}^{\theta+\pi/2} P(\sigma, \theta') d\theta' \tag{2.19}$$

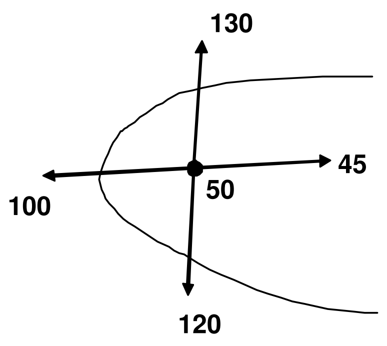


Figure 2.3: The direction with the highest difference in gray scale is not the direction with highest certainty. The direction from gray level 50 to 100 is the direction with highest certainty. The numbers in this figure corresponds to gray scale intensities.

If written in open form, this equation can be interpreted as a comparing the weighted difference of a center area of a disk to the half rings split at a certain angle (Figure 2.2(b)). The weightings over this circle would approximately look like a Laplacian of a Gaussian.

Now, let $E(\sigma, \theta)$ be the magnitude of filtering the image with the derivative of a Gaussian at the orientation θ. Based on this estimated direction, Edgeflow field is calculated as the vector sum:

$$\vec{S}(\sigma) = \int_{\theta-\pi/2}^{\theta+\pi/2} [\ E(\sigma,\theta')\cos(\theta') \quad E(\sigma,\theta')\sin(\theta')\ ]^T d\theta' \tag{2.20}$$

After this vector field is generated, the vectors are propagated towards the edges. The propagation ends and edges are defined when two flows from opposing directions meet. For both x and y components of the Edgeflow vector field, the transitions from positive to negative (in x and in y directions respectively) are marked as the edges. The boundaries are found by linking the edges. Segmentation is further pruned through region merging.

2.4 Graph Partitioning Methods (GPM)

Let us assume $G = (V, E)$ is a representation of an undirected graph, where V are the vertices and E are the edges between these vertices. V can correspond to pixels in an image or small regions (set of connected pixels). Image segmentation problem is then commonly formulated as the best bi-partitioning (or finding the best graph cut, $cut(A, B)$) of this graph based on a segmentation criterion. The segmentation criterion is either formulated as minimizing total pairwise similarities across a graph cut [57, 58, 2] or minimizing a certain cost (e.g. edge energies) along a cycle (a closed path, which also bi-partitions the graph) [59]. A graph partitioning method is usually represented by a pairwise similarity metric and a cost function defined on graph cuts.

Minimum cuts [60] define a very basic cost function:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \tag{2.21}$$

where $w(u, v)$ is the similarity metric. In [60], a similarity metric is defined as a decreasing function of edge strength between two neighboring pixels.

One problem with minimum cuts (also noted by its authors) is that it favors segmenting small but isolated regions since the cost is less if the boundary is shorter. One suggestion to prevent this is to have constraints on the size of the regions. Normalized cuts framework [57] offers a more elegant solution to this problem by introducing a way of normalization. The cost function for normalized cuts is:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \tag{2.22}$$

where $assoc(A, V) = \sum_{u \in A, v \in V} w(u, v)$ is the total similarity from nodes in A to all the nodes in the graph. $assoc(B, V)$ is similarly defined. The similarity function used is:

$$w(u,v) = \exp\left(\frac{-\|F(u) - F(v)\|}{\sigma_I}\right) \begin{cases} \exp\left(\frac{-dist(u,v)^2}{\sigma_X}\right) & \text{if } dist(u,v) < r \\ 0 & \text{otherwise,} \end{cases} \tag{2.23}$$

where F is some sort of a feature vector (intensity, color, texture) and $dist(u, v)$ is the Euclidean distance between nodes u and v. Another interesting property of normalized cut is its relation to normalized association.

$$\begin{aligned} Ncut(A,B) &= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}\right) \\ &= 2 - Nassoc(A,B) \end{aligned} \tag{2.24}$$

which means that minimizing normalized similarity across the cut is equivalent to maximizing normalized similarity within the regions.

Another cost function that propose normalization to the minimum cut problem is average cuts [58]. In average cuts, the normalization is done over the size of the regions respectively:

$$Acut(A,B) = \frac{cut(A,B)}{|A|} + \frac{cut(A,B)}{|B|} \tag{2.25}$$

A comparative analysis of minimum cut (with minimum size restrictions), normalized cut and average cut has been published recently [61]. The empirical comparison is done for the application of perceptual grouping of constant curvature segments. The results show no statistical advantage of one method over another. The results for all three methods are equally satisfactory, but a general fluctuation of performance depending on the image classes has been observed (aerial and indoor images show reduced performance). It is still possible that for the application of image segmentation, one method might perform better than the others.

Ratio cuts define two similarity metrics and two cost functions. The resulting cost function is then defined as the ratio of these two cost functions:

$$Rcut(A, B) = \frac{cut_1(A, B)}{cut_2(A, B)} \tag{2.26}$$

$cut_2(A, B)$ is usually selected as the length of the boundary between A and B, which means the total cost is normalized over the boundary size.

2.5 Other Methods

Another method that has been quiet popular within the last ten years is watershed segmentation [62-75], which uses morphological region growing to partition an image. Given a gradient information $g(x, y)$ about an image, the boundaries are found as the watershed lines if $g(x, y)$ is considered as a mountain landscape and if it is flooded at certain markers. Main advantage for watershed methods has been the simplicity and the computational efficiency. On the other hand, selecting the marker locations and the number of markers presents a challenge. If the markers are chosen as the local minima of $g(x, y)$ then an over-segmentation is inevitable. Region merging or multi-scale methods are usually utilized to create a better segmentation. In some ways, the region growing aspect of watersheds resembles the curve evolution process. Unfortunately watershed growing is not as flexible as curve evolution since growing in watersheds is only one dimensional, outwards. Another problem with watersheds is the difficulty to enforce constraints such as boundary smoothness. Most segmentation results that are shown to demonstrate watershed segmentation (especially after region merging) include very noisy boundaries. Recently Nguyen et al. in [73] offered a solution for this kind of problems by introducing constraints using energy minimization techniques. On the other hand this introduces extra computational complexity. While being actively studied, unfortunately convincing demonstration of segmentation results on natural images has not been achieved by watershed segmentation methods.

There has been also large amounts of work on texture segmentation in the last ten years [76-93]. Almost in all these works, segmentation results are demonstrated only on homogenous Brodatz-like textures [94]. Effectiveness of these methods on natural images has not been investigated.

Looking at the last five years of research in image segmentation, majority of the segmentation results that show convincing result on natural images have been based on graph partitioning methods that we discussed in Section 2.4, or methods that are extensions of these methods. Other than graph partitioning methods, one of the more interesting work is proposed by Tu et al. [95, 96]. This segmentation is based on a global optimization scheme called Data-Driven Markov Chain Monte Carlo (DDMCMC), which is formulated within the Bayesian framework. Many of the common segmentation criteria and processes such as edge detection, region growing, and split and merge, as well as models for regions and boundary has been used together to drive a generative process for best approximating the image from the existing data and models. Promising results on natural images are presented. Another data-driven segmentation technique is presented by Ren and Malik [97]. In this work, ground truth segmentations from Berkeley segmentation data set [29] are used as inputs to a learning system and classifier. By combining and merging regions from an over-segmentation of the image, final segmentation boundaries are found. Reasonable results on wide range of natural images has been demonstrated. While the work of Tu et al. approaches the segmentation problem from a generative perspective, the work of Ren et. al. tackles this problem from a discriminative perspective.

Chapter 3

Curve Evolution and Anisotropic Diffusion on Natural Images

In this chapter we develop optimization and evaluation frameworks for variational segmentation methods. While there is an extensive literature on image segmentation, including performance evaluation of several standard edge detection techniques, variational methods are relatively recent and are not evaluated on natural images. Much of the initial development of variational methods theory assumed simple image cases and it is not clear or obvious if those analysis hold good for natural images containing significant texture and clutter. In this context, we design, analyze and evaluate curve evolution and anisotropic diffusion schemes for image segmentation using a large collection of natural images. The work presented here is important for the understanding of the behavior of variational techniques, both their shortcomings and advantages, while dealing with real image data.

This chapter consists of two parts. In the first part we analyze curve evolution techniques, namely geodesic active contours, gradient vector flow and Edgeflow-driven curve evolution. We first describe a curve evolution framework that is based on Edgeflow vector field (Section 2.3.1). In this framework, the object boundaries are detected through evolutions of closed curves that are driven by an Edgeflow vector field. The motivation for using Edgeflow is that Edgeflow is experimentally shown to work reasonably well on thousands of natural images that are rich in color and texture whereas previous curve evolution methods target a limited class of images. We will use a modified version of Edgeflow as discussed in Section 3.1. Another novelty of this curve evolution is the integration of edge vector fields and the geometric curve evolution techniques. After demonstrating specific advantages of edgeflow on synthetic images, we evaluate several state of the art curve evolution techniques on natural images. We first optimize each curve evolution on a training set of images. On this data set of 20 images, the important parameters of each algorithm are tuned for best performance. Finally, these techniques are tested and their performance evaluated on 100 natural images. In these tests, Edgeflow-based curve evolution performs better than the state of the art edge based curve evolution techniques: geodesic active contours and gradient vector flow.

The second part of the chapter investigates anisotropic diffusion. The original Perona-Malik flow [6], self-snakes [50], and Edgeflow-driven anisotropic diffusion that we propose are compared. Each algorithm is optimized and their performances again are evaluated on natural images. Edgeflow-based anisotropic diffusion significantly outperforms both Perona-Malik flow and self-snakes.

3.1 Edgeflow-driven Curve Evolution

Edgeflow, as discussed in Section 2.3.1, describes a method for designing an edge vector field. An Edgeflow vector at point $p(x, y)$ is designed such that it estimates the direction towards the nearest boundary (See Fig. 3.3). Unlike standard edge detectors such as Canny's [55], the strength and direction do not represent the edge strength or orientation at $p(x, y)$–the vectors show more similarity to the second directional derivatives but are not based on the second derivative of the image. For example, due to the symmetry at the middle of a ramp edge, a point right on this edge will have zero length Edgeflow vector. On the other hand, classical methods such as Canny's find the edge strength (edgeness) and possible edge orientation at each pixel of the image thus creating two scalar functions, edge strength and orientation.

When designing the Edgeflow vector field, we make a modification to the original Edgeflow vector field such that the vector magnitudes in (2.20) are calculated using relative directional differences as opposed to directional derivatives. The original Edgeflow design calculated the direction and magnitude of the vectors independently using two unrelated techniques. We found this unnecessary and in our case, both direction and magnitude are calculated using relative directional differences. The equation (2.20) is changed to:

$$\vec{S}(\sigma) = \int_{\theta-\pi/2}^{\theta+\pi/2} [\ Error(\sigma, \theta') \cos(\theta') \quad Error(\sigma, \theta') \sin(\theta')\ ]^T d\theta' \tag{3.1}$$

Curve evolution and associated techniques are first used in other fields such as fluid dynamics before they found their applications in image processing. In [33], a general curve evolution is controlled by three separate forces:

$$\frac{\partial C}{\partial t} = \alpha\kappa\vec{N} + \beta F(x,y)\vec{N} + \gamma(\vec{S}\cdot\vec{N})\vec{N} \tag{3.2}$$

where $\kappa\vec{N}$ is a curvature-based force, $F(x,y)\vec{N}$ is a constant expansion/shrinking force, and $(\vec{S}\cdot\vec{N})\vec{N}$ is a force based on an underlying vector field whose direction and strength are independent of the evolving curve. α, β, and γ are constants and C is a curve. The curve is evolved in the normal direction by a combination of these forces. In our case we have the vector field and the curvature can be calculated. A simple curve evolution can be defined as

$$\frac{\partial C}{\partial t} = (\vec{S}_\sigma\cdot\vec{N})\vec{N} + \varepsilon\kappa\vec{N} \tag{3.3}$$

where $\vec{S}_\sigma$ is the Edgeflow vector field, ε is a constant weighting factor and κ is the curvature of the curve. The main purpose of the curvature term is to keep the curve smooth and to prevent the curve from splitting into many 1 or 2 pixel sized regions when passing through a noisy area. There are several problems we would face with this formulation. First of all, the curvature term smooths the curve and this would prevent the curve from capturing high curvature parts of the boundaries precisely. Second problem is that on a large flat area of the image, the vectors are of zero length. If the curve is initialized in a homogenous area, it will collapse under the influence of the curvature term and disappear. We will address these problems by reducing the effect of the curvature near the edges and we will add an expansion term to help the curve reach the object boundary. Fig. 3.1 illustrates how a curve can get stuck outside the reach of the vector field. For this kind of cases, (3.3) needs to be updated as we discussed.

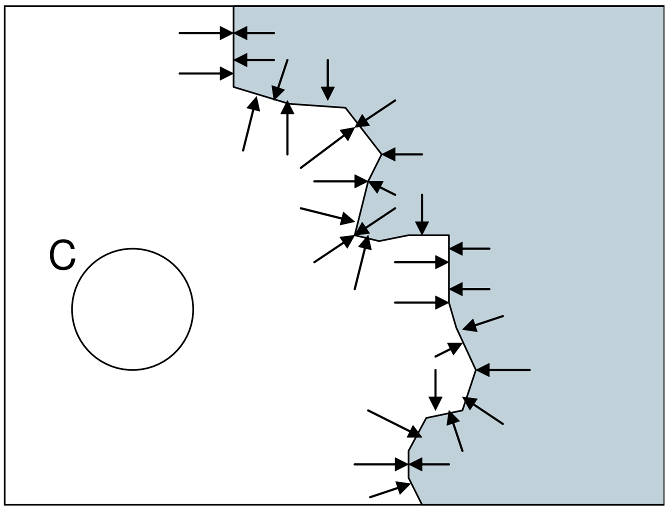


Figure 3.1: Illustration of the case where the curve is stuck in the background outside the reach of the vector field. The vector field is not able to pull the curve towards the edges.

Before addressing these problems, let us derive a scalar edge function from the Edgeflow vector field. As a natural choice, we define the edge strength function as the inverse gradient of the conservative component of the Edgeflow vector field. Using Helmholtz decomposition we can write Edgeflow as a sum of a conservative and a solenoidal vector field:

$$\vec{S} = \vec{S}_{con} + \vec{S}_{sol} = -\nabla V + \vec{\nabla} \times \vec{A} \tag{3.4}$$

where $\vec{S}_{con}$ is the conservative component and S_{sol} is the solenoidal component of $\vec{S}$, the Edgeflow vector field. Taking the divergence of both sides:

$$\vec{\nabla} \cdot \vec{S} = -\Delta V + \underbrace{\vec{\nabla} \cdot (\vec{\nabla} \times \vec{A})}_{0} \tag{3.5}$$

where Δ is the Laplacian. Since the second term is zero, we only need to solve a

Poisson equation [98] to find the edge function:

$$\vec{\nabla} \cdot \vec{S} = -\Delta V \tag{3.6}$$

To verify this solution, let us define the edge function $g = -V$ from a different aspect as a minimizer of the energy functional

$$\min_{V} E = \frac{1}{2} \int_U \|\vec{S} - \nabla V\|^2 \tag{3.7}$$

where $U \subset \mathbb{R}^2$ is a bounded and open set on which the image is defined. It can be seen that E is a function of ∇V. This problem can be solved by using the first variation of E. Let us define the Lagrangian $L = 1/2\|\vec{S} - \nabla V\|^2$. The Euler-Lagrange equation associated with (3.7) is then:

$$\frac{\partial \left(\frac{\partial L}{\partial V_x} \right)}{\partial x} + \frac{\partial \left(\frac{\partial L}{\partial V_y} \right)}{\partial y} = 0 \tag{3.8}$$

where $V_x = \partial V / \partial x$ and $V_y = \partial V / \partial y$. Derivation of this equation is given in the Appendix A. Note that $L(V_x, V_y) = 1/2(V_x - S^x)^2 + 1/2(V_y - S^y)^2$ where S^x and S^y are the components of $\vec{S}$. Straight evaluation of (3.8) results the same Poisson PDE (3.6), whose solution gives us the edge function V. This shows that the potential function V associated with the conservative component of Edgeflow is also the best we can achieve in the least squares (3.7) sense. See also [99] for a similar energy minimization approach where the inverse gradient is utilized for finding the level lines from color gradients.

Assume m is the mean and σ^2 is the variance of V. For contrast enhancement, values above $m + \sigma$ and values below $m - 6\sigma$ are clipped. After scaling to the interval $[0, 1]$, V has values around zero along the edges and values close to 1 on flat areas of the image. If we multiply the curvature term with V, this will reduce the smoothing effect of the curvature at the edge locations. This addresses one of the problems with the initial curve evolution we proposed. To address the other problem about evolving the curve in homogenous areas, we add a new term to the curve evolution, a constant force weighted by V. This term expands the curve in homogenous areas but its effects are reduced when the curve is close to the edges. After these additions, the curve evolution becomes:

$$\begin{aligned}\frac{\partial C}{\partial t} &= (\vec{S}\cdot\vec{N})\vec{N} + V\kappa\vec{N} + f(V)F\vec{N} \\ &= (\vec{S}_{sol}\cdot\vec{N})\vec{N} - (\vec{\nabla}V\cdot\vec{N})\vec{N} + V\kappa\vec{N} + f(V)F\vec{N}\end{aligned} \tag{3.9}$$

where F is a positive (negative) constant for expansion (shrinkage). We initially choose $f(V) = V$, but we will visit this selection later on in Section 3.3. These additions follow the ideas that are first proposed by Caselles et al. [37] and Malladi et al. [38]. Equation (3.9) can be implemented using level set methods [32, 33]. The equivalent level set equation for (3.9) is:

$$\frac{\partial U}{\partial t} = -\vec{S}\cdot\vec{\nabla}U + V\kappa\|U\| + f(V)F\|U\| \tag{3.10}$$

3.2 Comparison of Edgeflow with Geodesic Active Contours

Geodesic active contours introduced in [34, 36] have become the defacto standard in edge-based curve evolution. In this section we compare Edgeflow-based curve evolution (3.9) with the geodesic active contours (GAC) and point out the strengths of our method. GAC is a good method to compare because it is widely used and can be considered as the state of the art for edge-based curve evolution even though several variations of this method have been proposed recently. The GAC formulation is given by:

$$\frac{\partial C}{\partial t} = g(F + \kappa)\vec{N} - (\nabla g \cdot \vec{N})\vec{N} \tag{3.11}$$

where $g = 1/(1 + |\nabla \hat{I}_\sigma|)$ and $\hat{I}_\sigma$ is the Gaussian smoothed image. This definition of g introduces a nonlinear scaling that is aimed to enhance the weak edges. This evolution is derived, excluding the constant expansion term, as the local minimization of the total curve length weighted by g. Starting with an initial curve, a local minimum is searched so that the curve goes through the high values of the edge strength function. We will discuss the problems with this local minimization later in this section.

By comparison to GAC, first we show how Edgeflow vector field looks like and how the solenoidal component of Edgeflow is important for capturing concavities and high curvature sections of a boundary. Then we discuss the robustness of our method under noisy conditions. We demonstrate this on images at increasing noise levels.

Fig. 3.2 compares segmentation of a simple binary image using our method and geodesic active contours. The constant expansion term is multiplied with -1 so that the curve shrinks instead of expanding. The results demonstrate the importance of the solenoidal component of Edgeflow in extracting a precise boundary of the concavities. The spatial scale used in this example is $\sigma = 2.5$ and the image is of size 100×100. Fig. 3.2(d) shows that weighted curve length minimization has difficulty in handling concavities in the boundary [1]. When the curve reaches a small opening in the boundary, locally this is a minimum since going further increases the weighted curve length. On the other hand, going inside this opening results in a more optimal solution in the end.

Fig. 3.3 shows and compares the vector fields Edgeflow, $\vec{S}_{con} = \nabla V$, $\vec{S}_{sol}$, and $-\nabla g$. For visualization purposes, $\nabla \|\nabla \hat{I}\|$ is displayed instead. Due to the nonlinear scaling, $-\nabla g$ is not easy to visualize. On the other hand, the characteristics of these vector fields are similar. The vector fields are shown only in the neighborhood of the concavity. The main observation is that the solenoidal component of Edgeflow creates a force towards the inside of the opening. This additional information contained in Edgeflow is what is missing in geodesic active contours[2].

Another important characteristic for curve evolution is its noise sensitivity. For this purpose, we create ten images by adding noise to a binary image as shown in Fig. 3.4. These images, R1 to R10, are generated using the graphics software Paint Shop Pro's *Add Noise* dialog. Random noise at 100% setting is applied to the original image (Fig. 3.4(a)) iteratively ten times. Then, both our method and the geodesic active contours are applied to these noise-added images. The initialization of curve evolution is as shown in Fig. 3.2(a). For each image, the

[1]By increasing the weighting of the constant shrinkage term, the curve can be forced to evolve into the concavity. Note that the shrinkage term does not originate from energy minimization. But in our experiments GAC strongly favors not going into the concavity and we need to find specific weightings to achieve this. Later on we will see that when noise is added, playing with the weightings of the shrinkage term will not help. On the other hand, Edgeflow naturally evolves into the concavity pretty much for any selection of weightings.

[2]Advantages of using a flow consisting of a solenoidal component is also shown in [56]. This will be discussed more in Section 3.5.2.

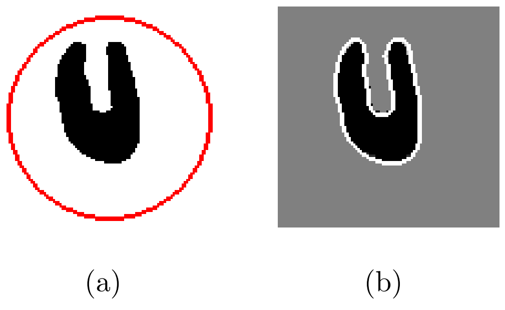
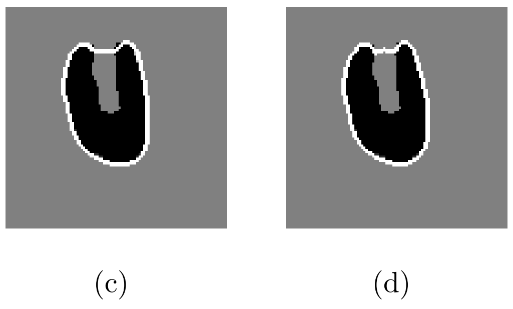



Figure 3.2: a) Initialization of the curve evolution. b) Segmentation result using Edgeflow. c) Segmentation using only the conservative component of Edgeflow. d) Geodesic active contour segmentation

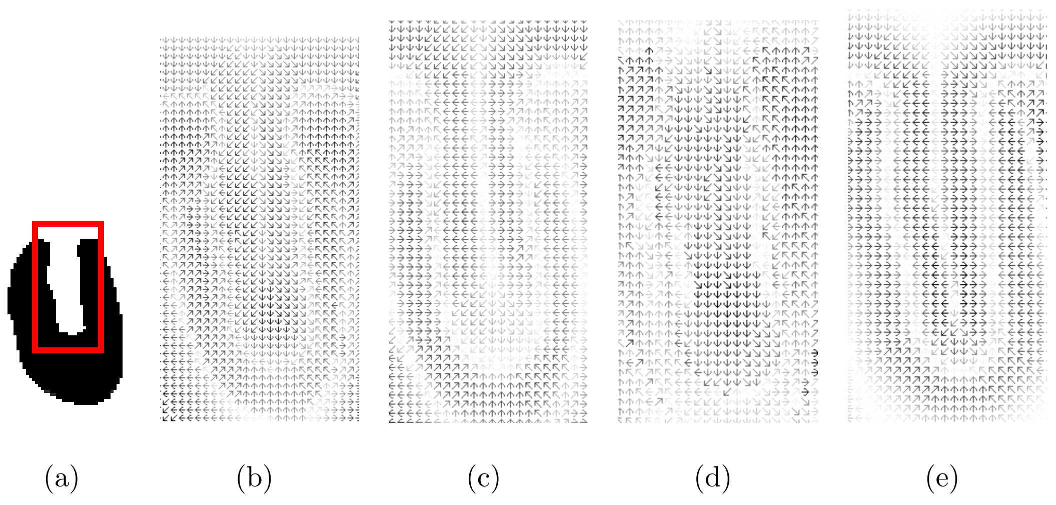
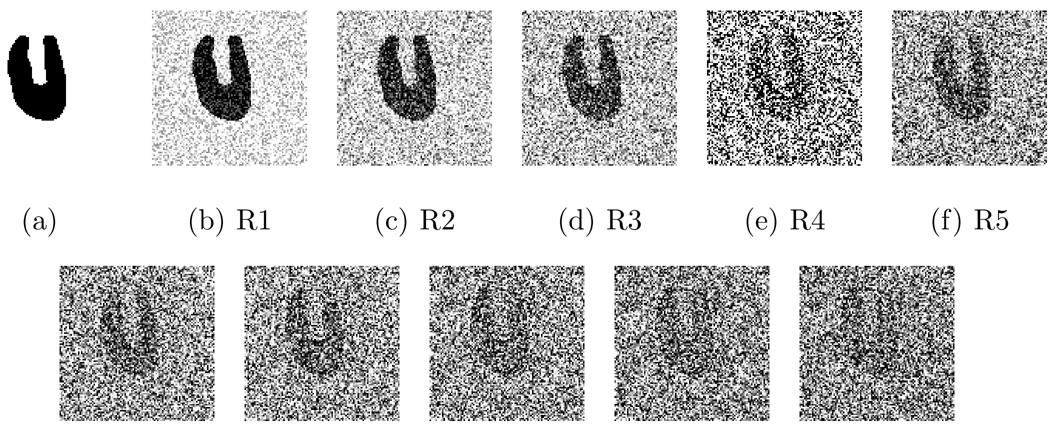


Figure 3.3: a) Binary image and the zoom rectangle, b) Edgeflow $\vec{S}$, c) conservative component of Edgeflow $S_{con} = \nabla V$, d) solenoidal Component of Edgeflow $S_{sol} = \vec{S} - \nabla V$, e) $\nabla \|\nabla \hat{I}\|$, similar to what is used in geodesic active contours.

weighting for the constant shrinkage term is manually adjusted for best results. We observe that GAC segmentation is able to segment the object in images R1 to R6, and the result for R6 is shown in Fig. 3.5(a). In R7 to R10 the curve collapses onto itself and disappears either because the boundary leaks at several edges or GAC completely misses the object. Edgeflow-based curve evolution on the other hand is able to capture the object with good precision for images R1 to R8. The segmentation for R8 is displayed in Fig. 3.5(c). Even on R10, our method is able to capture some parts of the object. This experiment demonstrates that our method is more immune to noise at both during the curve evolution and at the convergence to the object boundaries. In this experiment, we tried to create equal setting for both curve evolution techniques. $\sigma = 2.5$ pixels is chosen for the Gaussian smoothing. An interesting observation is that if nonlinear scaling is not used and the edge function is selected as $g = -\|\nabla \hat{I}\|$ and linearly scaled to the interval $[0, 1]$, the results are very similar to the ones where nonlinear scaling is utilized.

3.3 Improving Curve Evolution Stability using Divergence of Edgeflow

One of the main problems in edge-based curve evolution methods is boundary leaking due to the constant expansion term. To address this issue in our curve evolution framework, we suggest creating a binary mask around the edges. The purpose of this mask is to suppress the constant expansion term within the vicinity



(a) (b) R1 (c) R2 (d) R3 (e) R4 (f) R5

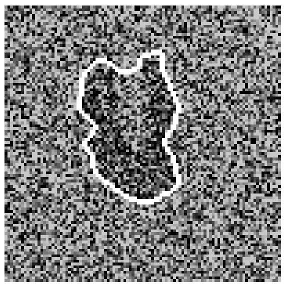
(g) R6 (h) R7 (i) R8 (j) R9 (k) R10

Figure 3.4: a) Original binary image. b-k) Images with random noise added increasingly.

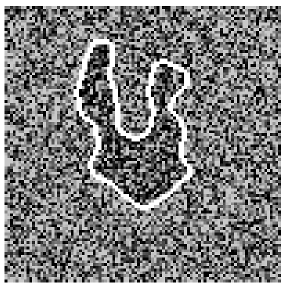
of the boundaries. We define an indicator function

$$\chi(x, y) = \begin{cases} \xi & \text{if } \operatorname{div}(\vec{S}) \leqslant 0 \\ 1 & \text{if } \operatorname{div}(\vec{S}) > 0 \end{cases} \tag{3.12}$$

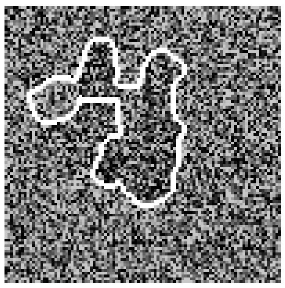
where ξ is a constant between zero and one. Note that this function only depends on the conservative component of $\vec{S}$ since $\vec{\nabla} \cdot \vec{S} = \Delta V$. χ is visualized in Fig. 3.6(d) for $\xi = 0$. As can be seen, this function takes the value 0 around the edges. For utilizing χ, we change $f(V)$ as $f(V) = \chi V$. Since the solenoidal component of Edgeflow has no effect in the calculation of χ, this scheme can be directly adapted by geodesic active contours. Fig. 3.6(b) shows the indicator function for geodesic active contours. Since divergence of $-\nabla g$ corresponds to the third derivative of the image, it is very sensitive to noise. In this case, suppressing the expansion term around the edges would mean suppressing it also around noise. In the case of GAC this would cause more harm than good by slowing convergence or causing


(a) R6 (Geodesic active contours)

(b) R6 (Canny)


(c) R8 (Edgeflow)

(d) R8 (Canny)


(e) R10 (Edgeflow)

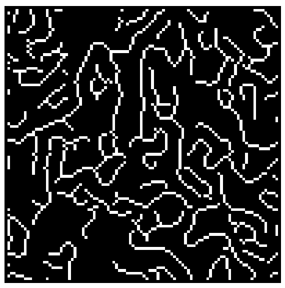

(f) R10 (Canny)

Figure 3.5: a) R6 segmented using geodesic active contours. b) Canny edge detection on R6. c) R8 segmented using Edgeflow-based curve evolution. d) Canny edge detection on R8. e) R10 segmented using Edgeflow-based curve evolution. f) Canny edge detection on R10.

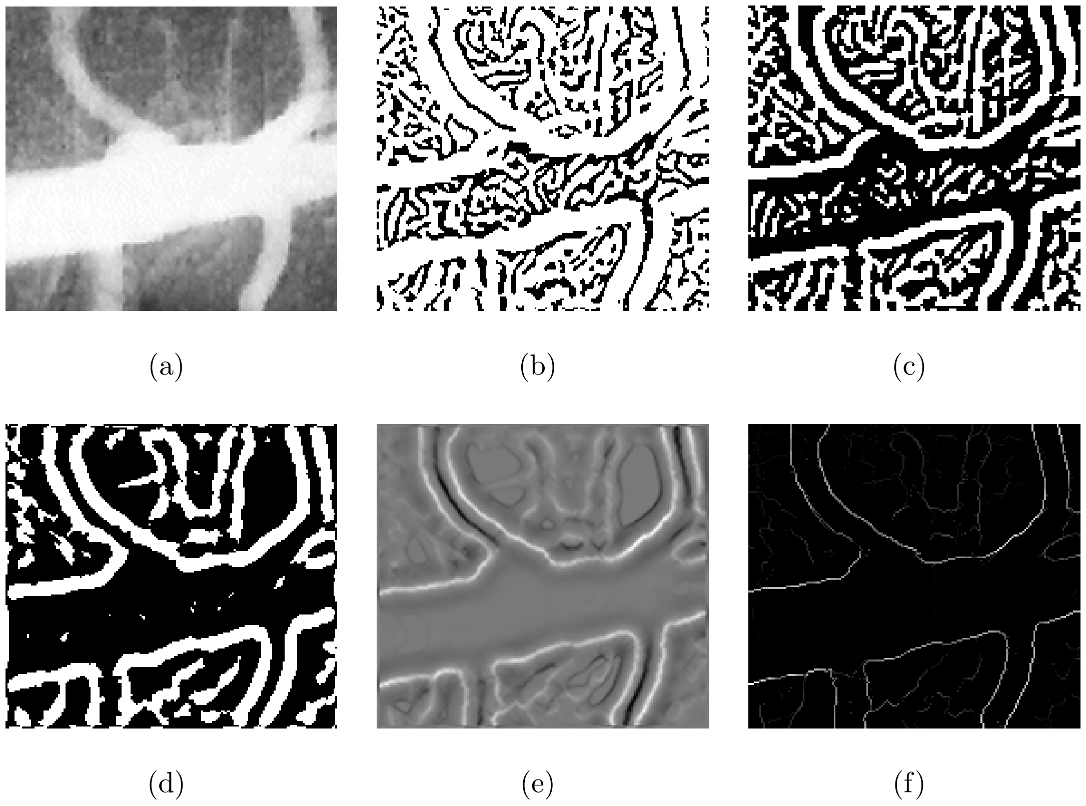


Figure 3.6: a) Blood vessel image; b) Divergence indicator function of $-\Delta g$ (white corresponds to low values, black to high values); c) Divergence indicator function of $\nabla \|\nabla \hat{I}\|$ (without nonlinear scaling); d) Divergence indicator function of $\nabla \cdot \vec{S} = \Delta V$; e) Divergence of $\nabla \cdot \vec{S} = \Delta V$; f) Edges detected using the original Edgeflow method (without vector propagation) [1].

curves to stop at false boundaries.

As is apparent from Fig. 3.6(d), χ can also be utilized to assist identifying best locations to instantiate the curves. For example, the boundaries of the black connected components in Fig. 3.6(d) or their convex decompositions are good candidates to instantiate curves close to the final object boundaries.

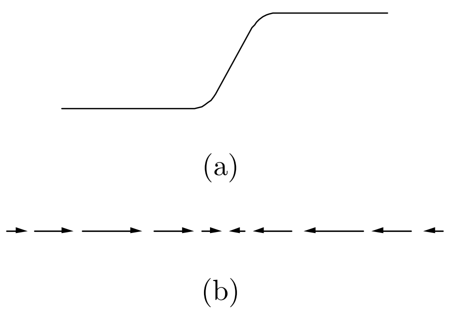


Figure 3.7: a) An edge profile, b) ∇V

3.4 Berkeley Segmentation Ground Truth

In this section, we will optimize and compare three edge-based curve evolution techniques on natural images. The techniques we evaluate are: geodesic active contours (GAC) [34], gradient vector flow (GVF) [56], and Edgeflow-driven curve evolution (EF).

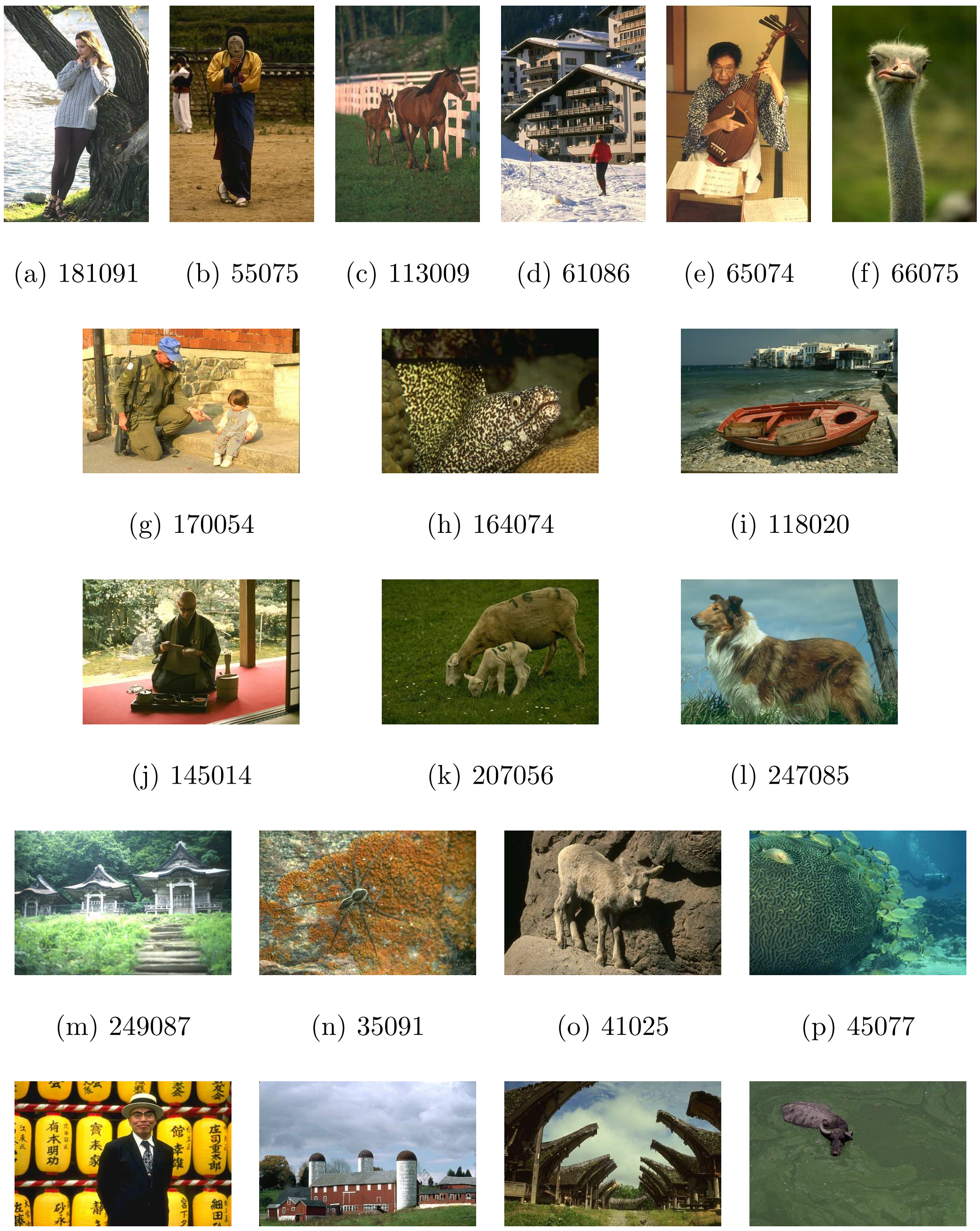
The experiments for the evaluation are conducted on a data set that is provided by UC, Berkeley Computer Vision Group [29] and can be downloaded from http://www.cs.berkeley.edu/projects/vision/grouping/segbench/. Images are in JPEG format and the segmentation ground truth as well as accompanying source code for many useful tasks such as matching a segmentation result to a ground truth are included. The data set is divided into two sets: training and test sets. The suggested training set consists of 200 images and the test set consists of 100 images. The idea is to optimize all the parameters of your algorithm on the training set and then the evaluation is done on the test set. Ideally a more thorough evaluation such as cross validation is desired, but unfortunately the computational needs for this is extensive and not practical even on the high-end computers we have currently. Actually we reduce the training set to 20 images by random sampling for computational reasons. The selected training images are given in Fig. 3.8. Visually, the images in our set seem to be diverse. The original images are of size 481x321. For our experiments, we resize these images to 192x128 for computational reasons.

The ground truth associated with the images consists of at least 5 or more manual segmentations for each image. Each of these manual segmentations are assumed to be correct and a result from an automated segmentation is matched to each of these manual segmentations separately. Matching a segmentation to a ground truth requires a definition for edge localization. Let D be the length of the diagonal of an image. To count as a match, the maximum distance of an edge to a ground truth edge should be less than $\frac{0.75}{100}D$. Bi-partite graph matching is used to prevent more than two edges to match the same ground truth edge. More details can be found in [24].

We conduct our experiments in this chapter using gray scale values of the images. The main reason for this decision is simplicity. Moreover, the curve evolution methods we evaluate are mainly proposed and demonstrated for gray scale intensities (except that original Edgeflow method's main target was texture features).

Why compare with color ground truth

The Berkeley segmentation data set provides two sets of ground truths. The first data set is created by displaying the original color images to human subjects



(a) 181091 (b) 55075 (c) 113009 (d) 61086 (e) 65074 (f) 66075

(g) 170054 (h) 164074 (i) 118020

(j) 145014 (k) 207056 (l) 247085

(m) 249087 (n) 35091 (o) 41025 (p) 45077

(q) 65019 (r) 97017 (s) 76002 (t) 80099

Figure 3.8: Training set of images and their IDs in Berkeley segmentation data set.

and having them to generate the ground truth segmentations manually. The second data set is created as follows: First, the quality of color images are reduced by converting them to gray scale. These gray scale images are then shown to the subjects and corresponding ground truth segmentations are generated (another ground truth is created by showing inverted gray scale images but this is not of interest to us). Based on the availability of these two data sets, an important question about which ground truth data set we should choose to use remains. The creators of the segmentation data set suggests using gray scale ground truth for evaluating methods that are based on gray scale intensities (or texture) and color ground truth for methods that utilize color information. Their work on evaluating edge detection methods [24] also follows this methodology. Unfortunately performance measures calculated on two separate data sets cannot be compared. Without proper normalization, such comparisons do not have any statistical meaning. On the other hand, normalizing these performance results is close to impossible considering the complexity of the process and countless unknowns that are associated with the human visual system and our knowledge about the semantics of the objects in the real world. Besides, these two sets of ground truth are generated by different people. This makes it more difficult to compare a method that is evaluated on gray scale ground truth versus a method that is evaluated on color ground truth. Even if both ground truth is created using same type images (both color or both gray scale), the fact that the ground truth are created by different people makes it hard to draw any useful comparisons across data sets. Performance comparisons can only be meaningful if experiments are conducted under the same external conditions.

Considering that the natural images are acquired in color, the meaning of gray scale intensities can be thought of as one of the following: 1) Conversion of color images to gray scale can be seen as 3 to 1 lossy compression or dimensionality reduction from three dimensions to one. 2) We can consider brightness as an image feature that the segmentation methods might like to use. Neither of these cases warrant that we utilize a ground truth that is generated from brightness features alone. If an algorithm favors using dimensionality reduction, it should be clear that some information is lost and this will be reflected in the final performance–gains in terms of computational efficiency is possible but we are not measuring this. If brightness is an image feature (and certainly it is), then should we really create separate sets of ground truth for each image feature? (e.g. for evaluating an algorithm that analyzes R, G, and B components of a color image separately and combines these results to generate a segmentation, should we create separate ground truth for R, G, and B images and combine them at the end?)

For people who are not familiar with the ground truth data from the Berkeley segmentation data set (BSDS), the contents of this segmentation ground truth might not be clear. Previous ground truth data sets [25] are created mostly to simulate the computer's view of the images and semantics about the objects are avoided (or not as much used). On the other hand, BSDS ground truth is created by people unfamiliar with the segmentation research and semantics of the objects are heavily utilized (Very obvious edges and boundaries are consistently avoided if they do not contain any semantic meaning with respect to the objects). One might wonder why there are so many differences between the color ground truth and gray scale ground truth since both of them correspond more or less the same scene. Our subjective comparison of the ground truth show that there are two main reasons for this:

- When color images are converted to gray scale, some regions emerge, which are not easy (or impossible in some cases) to be recognized as to what they correspond to. Whenever the subjects cannot figure out what the region might correspond, they marked it as a segment. On the other hand, the same region might correspond to some illumination effect, for example, shadows. Using color images, these regions are usually not labeled as separate segments.

- Some boundaries that are very visible become very low contrast when converted to gray scale. This causes confusion since certain low levels of contrast (especially if the neighborhood has higher contrast) is not easyly parsed by the human visual system. On the other hand, this might not be a problem for a computer algorithm.

The conclusion we reach based on our observations is that there are two sets of very useful ground truth data that comes with BSDS and each can be used to evaluate *any* segmentation method. On the other hand we would like to note that one of them is clearly of higher quality. For this reason we choose to use the *better* ground truth data (ground truth from color images) in our experiments.

3.5 Evaluation Framework

We can formulate a generic edge-based curve evolution with the following PDE:

$$\frac{\partial C}{\partial t} = \alpha F_1 + \beta F_2 + F_3 \tag{3.13}$$

where F_1 is the term for constant expansion weighted by an edge function, F_2 is the curvature-based force, and F_3 is the vector field based force. There are three parameters that directly affect the curve evolution, α, β and the scale σ at which the edge function and the vector field are generated (usually σ corresponds to the variance of the Gaussian smoothing). See [100] for a discussion about the effects of changing α and β on the segmentation results (the curve evolution in [100] does not contain F_3). β, the weighting factor for the curvature-based term adjusts the smoothness of the curve during its evolution. If β is selected too high, the curve will be inflexible and would favor staying in straight or circular shape. If β is too small, then the curve become noisy and unstable, and potentially may split into many small curves. On the other hand, if β is adjusted to a reasonable value, variations in β will not have a significant effect on the final segmentation. Reasonable value for beta is decided by considering the following conditions: 1) the curve's shape is not enforced by the curvature term, 2) the curve can take any shape depending on the edges in the image, and 3) the curve stays smooth during the evolution. Based on these observations, we fine tune β manually on each curve evolution technique to a reasonable value. Visually, we try to keep the effect of the curvature same for each method we are testing.

The training phase requires finding the optimum values of α and σ for each of the curve evolution techniques. We suggest and utilize some form of combinatorial optimization for the optimization of these two parameters. The scale σ is directly linked with the edge detection and edge localization quality. Another important feature of σ is that the reach of the vector field becomes larger with increasing σ. The reach of the vector field has an effect on carrying the curve to the edges and preventing boundary leaks. On the other hand, α is a very important parameter in terms of helping the curve reach the desired boundary. A key behavior of the curve evolution is that the constant expansion term (weighted by the edge function) most of the time competes with the vector field to decide if the curve converges to a boundary or not. If the strength of the vector field is larger than or equal to the strength of the expansion term (along the whole boundary), the curve stays at the boundary[3]. If α is selected too high, the curve will not stop at the edges, and if α is too low, the curve might not reach the boundary.

Given a scale σ, we first generate the edge function and the vector field. Then, a useful range of values for α is identified. The range of values for α should start from a low value which is not high enough to carry the curve to the boundaries (e.g. 0) to a high value for which the curve does not stop at any edge point. This interval for α depends on the curve evolution technique, but it does not change from one scale to another in our experiments. Once we decide on the interval for α, this interval is sampled at equally spaced points. For each of these values of α, the curve evolution and the associated segmentation are generated.

A measure of goodness of segmentation is needed to compute the optimal

[3]This is a simplification of the process. Certainly curvature-based force and the alignment of the curve's normal vector and the vector field play important roles on the convergence.

value of alpha. In [24], a way to match a segmentation to the ground truth and calculating precision and recall for the match is proposed. First, morphological thinning is applied to the binary segmentation map. Then the thinned edges are matched to the ground truth. Based on this match, precision P–number of correct matches divided by total number of edges detected and recall R–number of correct matches divided by total number of edges in the ground truth, are calculated. Ideally a high value for both precision and recall is desired, but in practice one value of α would give higher recall, while another value of α might result in a better precision. To define a single goodness measure from precision and recall, the F-measure has been used [101]:

$$F = \frac{P \cdot R}{\gamma P + (1-\gamma)R} \tag{3.14}$$

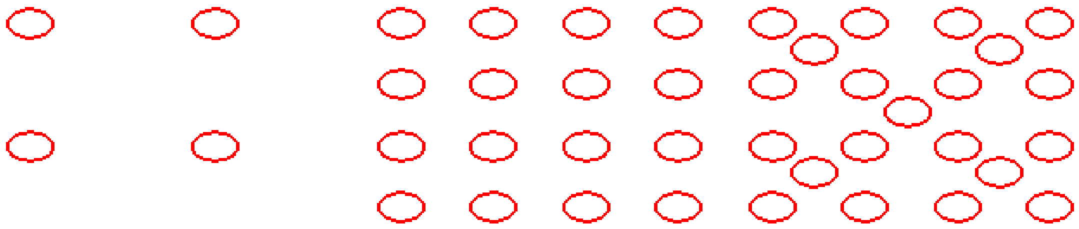
where γ defines the tradeoff between precision and recall. In [24], γ is selected as 0.5 (precision and recall are considered equally important). A better way of deciding on the optimum γ for image segmentation task would be through subjective experimentation where for a range of γ, human subjects are shown several segmentations with equal F-measures and asked if the segmentations are equally good. However, we are not aware of any such subjective tests reported in the literature. For simplicity and for compatibility with the other work on this data set, we also set γ to 0.5 in our experiments. For better resolution, successive samples of α, α_n and α_{n+1} are interpolated and P and R are estimated by linear interpolation using the P and R values at α_n and α_{n+1}. Finally, the value of α with the highest F-measure is selected as the optimum for this scale. This process is repeated for a range of scales, starting with $\sigma = 0.25$ pixels going up to $\sigma = 3.75$

pixels. The optimum value for α and the corresponding F-measure are found for each of these scales and the scale with the highest F-measure is selected as the optimum scale.

Another question regarding F-measure is that how much of a difference in F-measure corresponds to a visible improvement and how much of a difference is statistically insignificant. There is no clear answer to this. Subjectively, an improvement of 0.01 is visibly better. We report our experimental results up to the third decimal digit.

Curve evolution methods are not proposed and are generally not intended for a full segmentation of an image. A single curve is not enough for capturing all boundaries in an image. For this purpose, we propose the following methodology for generating an image segmentation using multiple curves. First, four multi-part curves, each with four sub-parts, are initialized on the image in a grid fashion (Fig. 3.9 (a) and (b)). The main feature of a multi-part curve is that the sub-parts of a multi-part curve would merge if they touch each other. While multi-part curves behave like multiple curves without extra computational complexity, they can not capture the T-junctions. In addition to these 4 curves, 5 more single-part curves are also instantiated (See Fig. 3.9 (c)) making it a total of 9 curves. We need to note that each extra curve adds to the computational complexity and therefore it is unpractical to use a large number of curves. This creates a tradeoff between the high edge detection rate and computational complexity. We have found that the proposed setup works quiet well.

To decide if a curve has converged or not, we need to define a convergence



(a) (b) (c)

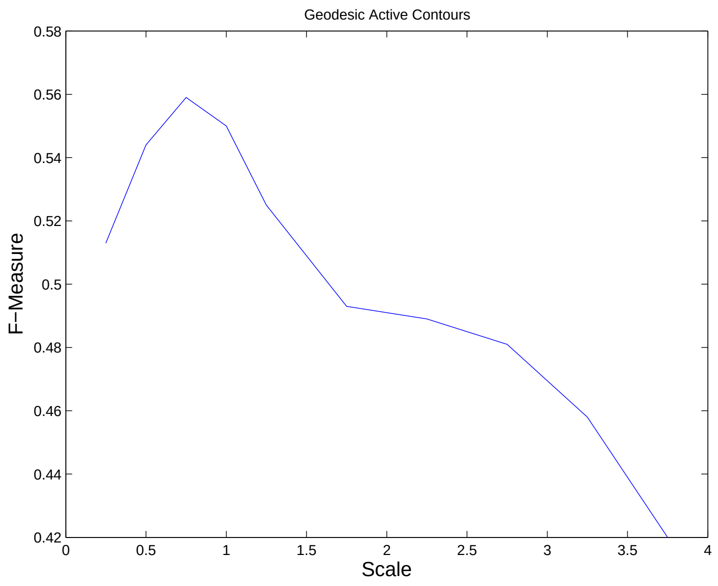
Figure 3.9: Curve instantiation setup. a) A multi-part curve, b) 4 multi-part curves each with 4 sub-parts, c) 9 curves total: 4 multi-part curves and 5 single-part curves.

criterion. Our convergence criterion is closely linked with our implementation of the curve evolution. In this work, curve evolution is implemented using narrow band level set methods (Section 2.2.4). We take the narrow band size as six pixels. If the curve stays within this narrow band for 60 iterations, we conclude that the curve has converged. As a failsafe, after 8000 iterations the curve is stopped regardless.

It is to be noted that our primary goal is not to do a comparison of curve evolution versus other segmentation techniques. Our curve evolution setup including the number of curves instantiated and their locations we chose might be suboptimal. On the other hand, our objective is to analyze various edge-based curve evolution techniques under the same conditions.

3.5.1 Training Geodesic Active Contours (GAC)

By the design of GAC, the range of edge stopping function g is nonlinearly scaled between 0 and 1. By visual inspection, we set β (weighting factor for the



(a)

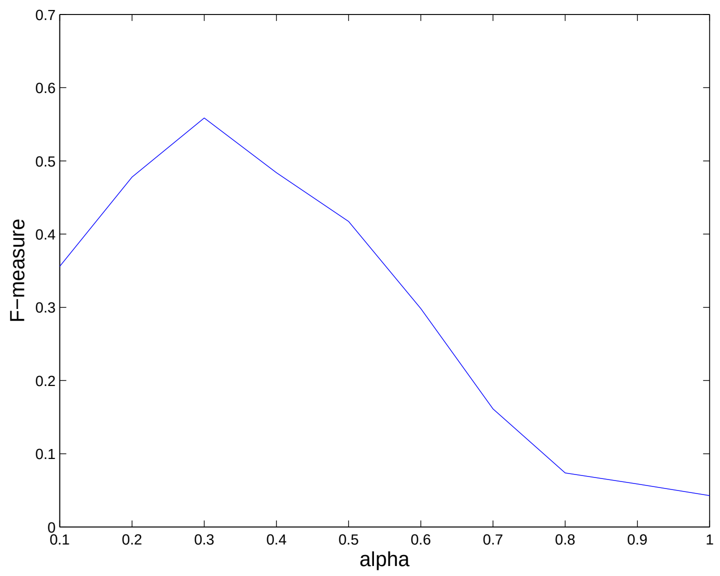
Figure 3.10: Performance of Geodesic Active Contours on the training set.

curvature-based term) to 0.3, and the range of α is chosen from 0.1 to 1 with increments of 0.1. The scales that we experiment are $\sigma = 0.25$, 0.5, 0.75, 1, 1.25, 1.75, 2.25, 2.75, 3.25, 3.75.

Fig. 3.10 shows the performance of GAC with the change of scale. The performance peaks at $\sigma = 0.75$ ($F = 0.559$). Fig. 3.11 shows the performance behavior with respect to α at $\sigma = 0.75$ for the training set. The optimum is at $\alpha = 0.3$. For giving a clearer picture about the evaluation process, specific performance results on individual images are given in Table 3.1. We don't show these individual results for other methods since they don't have a direct effect on the final performances.

Image ID	Optimum α	Recall	Precision	F-measure
113009	0.3	0.514059	0.822785	0.632774
118020	0.3	0.50296	0.686286	0.580493
145014	0.3	0.514466	0.656368	0.576818
164074	0.2	0.381296	0.279054	0.32226
170054	0.3	0.618361	0.597936	0.607977
181091	0.4	0.587639	0.693153	0.636049
207056	0.3	0.613432	0.586792	0.599816
247085	0.3	0.432552	0.553138	0.485469
249087	0.4	0.54278	0.47112	0.504418
35091	0.3	0.389957	0.549455	0.456166
41025	0.4	0.663312	0.575142	0.616088
45077	0.3	0.372615	0.49919	0.426714
55075	0.3	0.518105	0.785278	0.624308
61086	0.5	0.551292	0.497112	0.522802
65019	0.5	0.686889	0.891313	0.775862
65074	0.4	0.605895	0.817382	0.695926
66075	0.4	0.471123	0.785714	0.589047
76002	0.4	0.637881	0.715946	0.674663
80099	0.27	0.326446	0.277921	0.300235
97017	0.3	0.552249	0.652571	0.598233

Table 3.1: Optimum values for each image at $\sigma = 0.75$ using GAC



(a)

Figure 3.11: Performance with the change of α for $\sigma = 0.75$ (GAC)

3.5.2 Training Gradient Vector Flow (GVF)

Gradient Vector Flow (GVF), which is proposed by Xu and Prince [56], defines a method of generating a vector field such that the vector field flows towards the edges that are contained in the image. Given an edge strength function $f(x, y)$ of the image (for example $f(x, y) = \|\nabla I\|$), this vector field $\vec{G} = [u(x, y)\ v(x, y)]$ is realized as the minimization of the following energy functional:

$$E = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\vec{G} - \nabla f|^2 dxdy \qquad (3.15)$$

This can be interpreted such that the flow is equal to ∇f where $|\nabla f|$ is large enough and in homogenous areas where $|\nabla f|$ is small, the flow is smoothly interpolated from ∇f. While the reach (effective area) of ∇f is only around the edges, GVF extends the reach of the vector field to a larger area. The components of GVF are calculated by solving the following equations:

$$\mu\Delta u - (u - f_x)(f_x^2 + f_y^2) = 0 \tag{3.16}$$

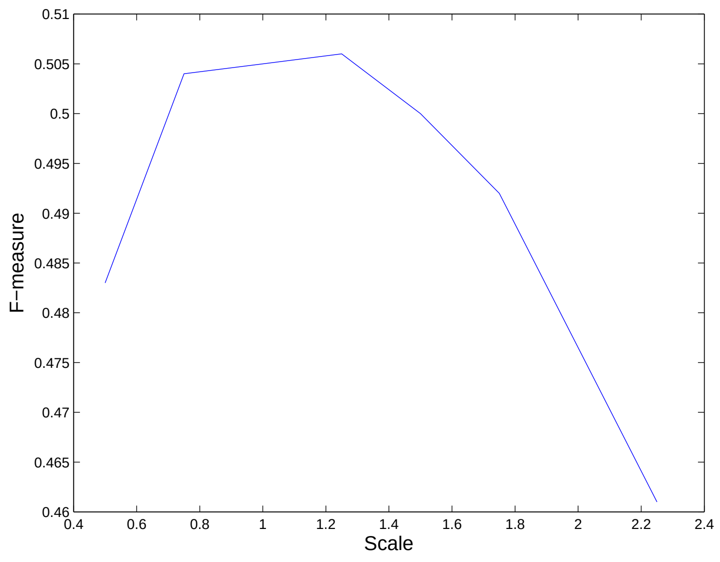
$$\mu\Delta v - (v - f_y)(f_x^2 + f_y^2) = 0 \tag{3.17}$$

GVF is initially proposed to be used as an external force field within the parametric curve evolution (snakes) framework. Later on use of GVF within geometric curve evolution framework is also proposed by Xu et al. [35] and Paragios et al. [102]. Variations of GVF as well as the idea of normalizing the vector field to unit length vectors is proposed and used in [103, 102].

Matlab source code for calculating GVF has been provided at the web site of its authors Xu and Prince (http://iacl.ece.jhu.edu/projects/gvf/). Our implementation directly follows this source code. The decisions for parameter settings are done based on the suggestions of the authors in their paper [56] and in their source code. μ is set to 0.1 and the number of iterations is set to $\sqrt{NM}$ where N is the width and M is the height of the image.

The curve evolution setup for GVF is exactly same as the one for GAC except that the edge stopping function and the vector field are different. The edge strength function is selected as $f = \|\nabla I\|$ and f is linearly scaled to the interval $[0, 1]$. Edge stopping function is calculated as $g = -f + 1$. β is visually set to 0.003, and the range of α is from 0 to 0.1 with increments of 0.01. The scales used are $\sigma = 0.5$, 0.75, 1, 1.25, 1.75, and 2.25.

Fig. 3.12 shows the performance of GVF with respect to σ. Performance peaks at $\sigma = 1.25$ for $\alpha = 0.02$ ($F = 0.506$). While we expected the opposite,



(a)

Figure 3.12: Performance of GVF on the training set.

unfortunately the performance of GVF is much lower than GAC on the training set (0.51 vs 0.56). The vector field for GVF works well as expected when there are clear step edges in the image. On more complex images (which have weak edges along the boundary and clutter) and for small scales (for which ∇f might be thin along the edges), GVF is not able to interpolate the vector field as well. In some cases weak edges are eliminated from the vector field after diffusion process. As previously discussed, if a large enough part of the boundary is not captured by the vector field, the whole boundary is usually missed by the curve evolution.

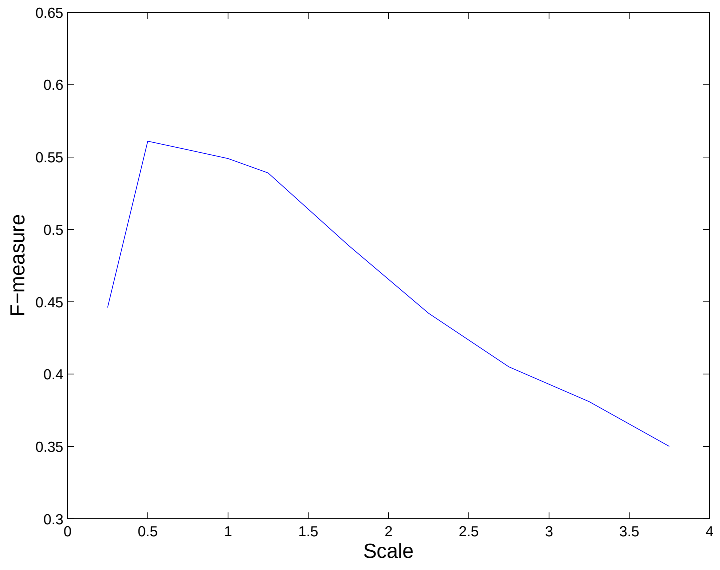
3.5.3 Training Edgeflow (EF)

Considering that the design of EF is not fixed like GAC or GVF, while we optimize EF, we also test different design decisions using the training set. The first decision we need to make is about the scaling and the dynamic range of the edge function V and the vector field $\vec{S}$. One option is to scale them independently, and the other option is to scale them together by the same factor.We will investigate both options. First we conduct the experiments for independent scaling. Values of V and magnitudes of $\vec{S}$ are both linearly scaled to the interval $[0, 1]$. Then both of them are divided by their means for normalization. For this setup we select β as 0.3 and the range of α is set as 0.1 to 1 with increments of 0.1.

Fig. 3.13 shows the performance of EF on the training set. The graph peaks at $\sigma = 0.5$ and $\alpha = 0.2$ ($F = 0.561$). The performance is about the same as GAC.

In Section 3.3, we introduced the use of the divergence of Edgeflow for improving the performance of our curve evolution. To verify our claims, we conducted experiments for $\xi = 0.5$ and 0.75. The results can be seen in Fig. 3.14. The results does not show significant improvements, only a slight improvement for $\xi = 0.75$ ($F = 0.562$ at $\sigma = 0.75$).

An interesting observation about these results is that despite showing slightly better performance than GAC, when compared to GAC, EF's performance falls much more rapidly with respect to scale increase. The problem originates from the use of Gaussian offset at 4σ. This large Gaussian offset introduces poor localization in EF with increasing scale. Based on this observation, we redesigned a new Edgeflow vector field using a Gaussian offset of σ. We call this vector field

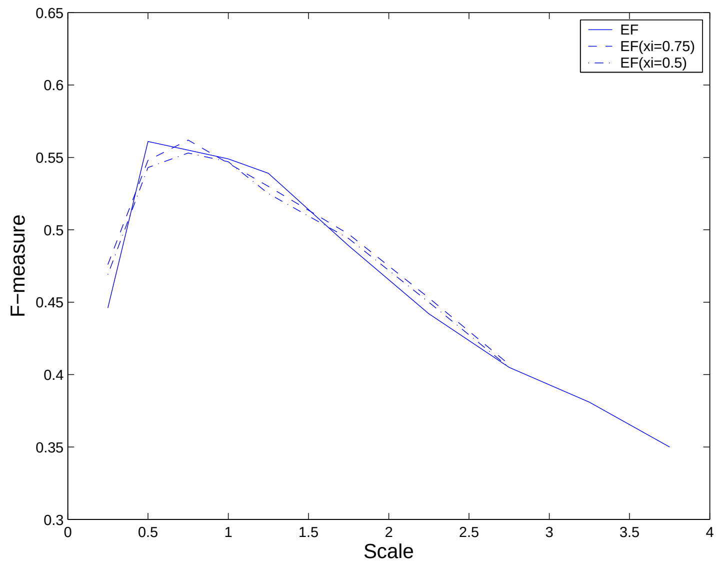


(a)

Figure 3.13: Performance of EF on the training set.

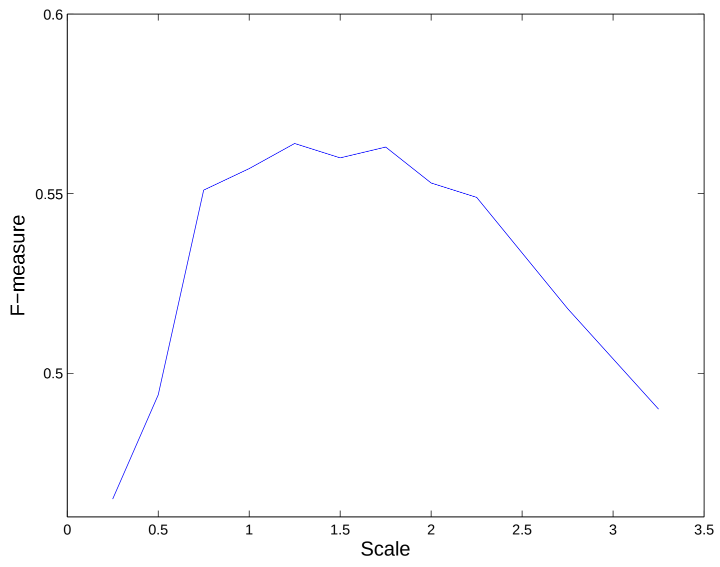
and the associated curve evolution as EF2. Fig. 3.15 plots the performance of EF2. It shows that EF2 has a better performance ($F = 0.564$) and the performance is stable across a wide range of scales. Comparison of EF2 versus EF along with GVF and GAC is given in Fig. 3.16.

We now repeat the experiments for EF and EF2 by scaling the edge function V and the Edgeflow vector field in a dependent way. We first calculate V by solving the Poisson equation (3.6). Let $m = Min(V)$ and $M = Max(V)$. We subtract m from V to make the minimum 0. After that both V and x and y components of $\vec{S}$ are divided by $M - m$. Then we multiply $\vec{S}$ by 20 so that α and β parameters can be kept same as in other experiments. We call the corresponding curve evolutions EF$'$ and EF2$'$. As shown in Figures 3.17 and 3.18, dependent scaling improves



(a)

Figure 3.14: Performance comparison for $\xi = 0.5,\ 0.75,\ 1$



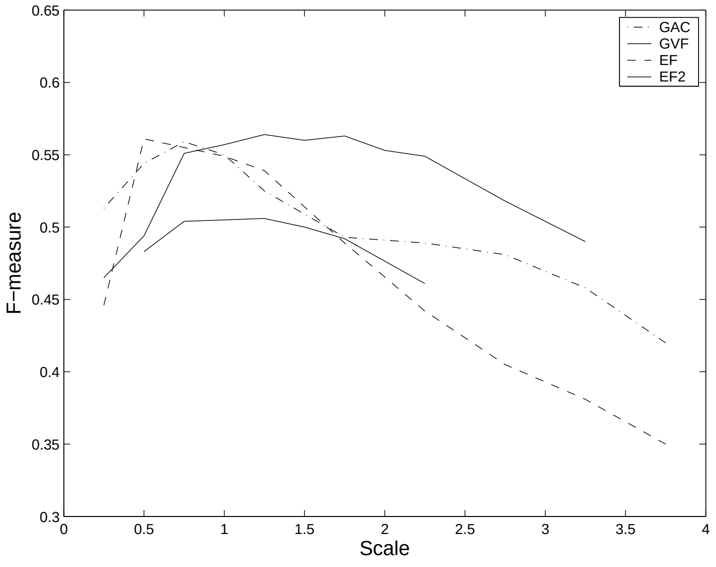
(a)

Figure 3.15: Performance of EF2 on the training set.

the performance ($F = 0.566$ for EF$'$ and $F = 0.573$ for EF2$'$). Fig. 3.19 shows the results for EF$'$, EF2$'$, GAC and GVF.

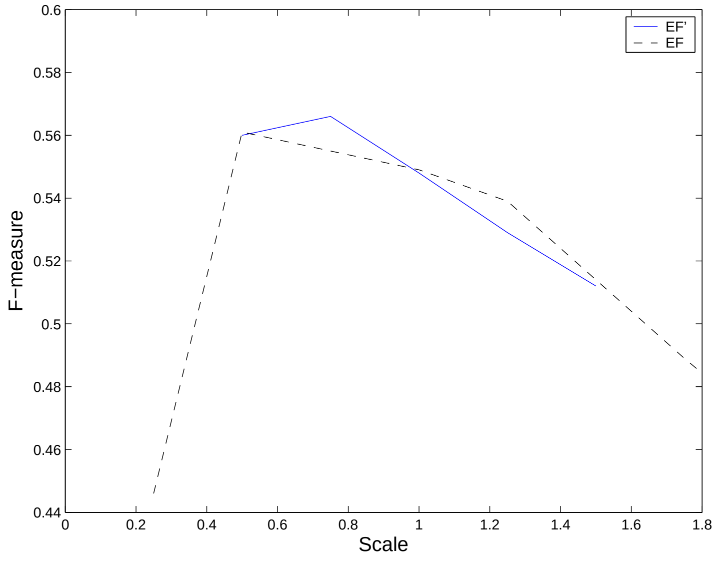
3.5.4 Test results

We apply curve evolution techniques we trained so far to the test set of 100 images associated with the Berkeley segmentation data set. For edgeflow, we test both EF$'$ and EF2$'$. A summary of the optimum values that are calculated form the training set, the performance on the training set, and the performance on the test set are provided in Table 3.2. GVF perform much worse than other techniques as expected based on the training phase. The results show that both EF$'$ and EF2$'$ show about 0.02 improvement with respect to GAC. More importantly, while GAC



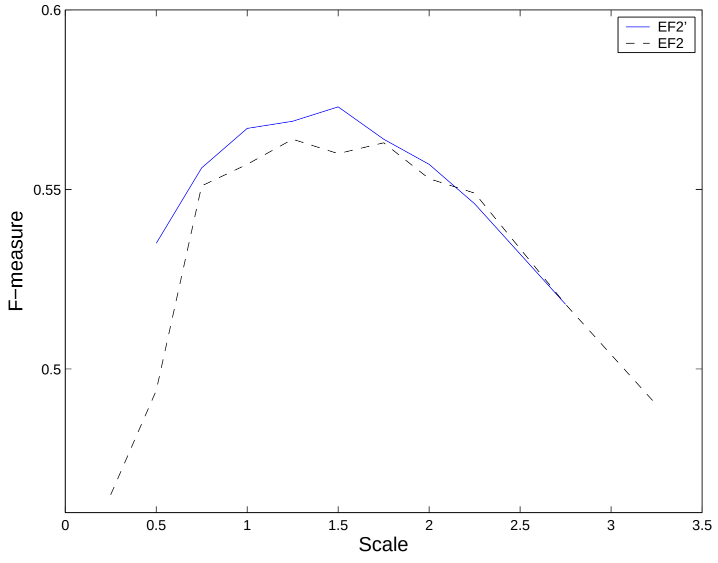
(a)

Figure 3.16: Performance comparison of EF, EF2, GAC and GVF on the training set.



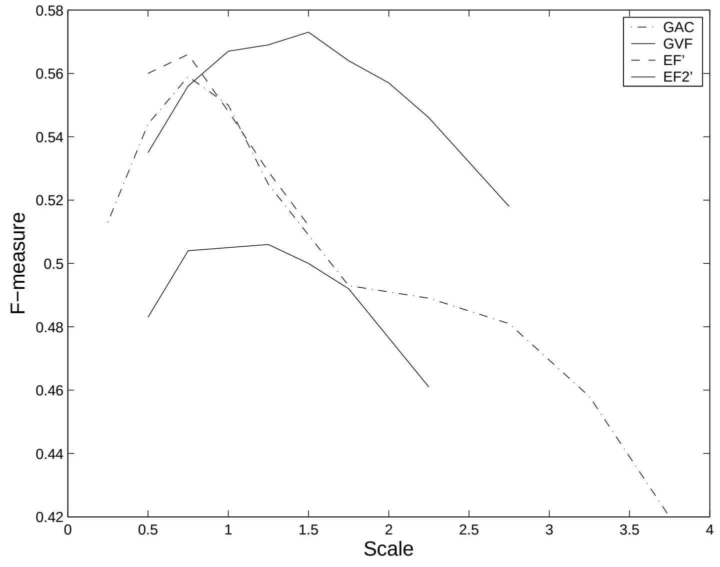
(a)

Figure 3.17: Performance comparison of EF and EF′.



(a)

Figure 3.18: Performance comparison of EF2 and EF2′.



(a)

Figure 3.19: Performance comparison of EF′, EF2′, GAC and GVF on the training set.

Technique	σ	α	β	F (on training set)	F (on the test set)
GAC	0.75	0.3	0.3	0.559	0.511
GVF	1.25	0.02	0.003	0.506	0.474
EF$'$	0.75	0.3	0.3	0.566	0.525
EF2$'$	1.5	0.3	0.3	0.573	0.526

Table 3.2: Summary of the optimum values for curve evolution

has a peak performance at a specific scale, EF2$'$ has been shown to be less sensitive to scale changes and performs quite well over a large range of scales.

Fig. 3.20 shows select segmentation results for each of the curve evolution technique on the test images. From these results it can be seen that GAC is more aggressive in capturing a large number of boundaries, which sometimes causes over-segmentation. The reason for this behavior is related to the nonlinear scaling of the edge function that is aimed to enhance weak edges. Visual evaluation has been the standard way of evaluating segmentation in the past. On the other hand, these results show that visually evaluating the segmentation quality is not easy and systematic evaluation techniques are necessary.

3.6 Anisotropic Diffusion

Anisotropic diffusion is another important variational segmentation method. The main idea in anisotropic diffusion is to smooth the homogenous areas of the image while enhancing the edges, thus creating a piecewise constant image from which the segmentation boundaries can be easily obtained. Anisotropic diffusion was first proposed by Perona and Malik [6]. Previous work on anisotropic diffusion

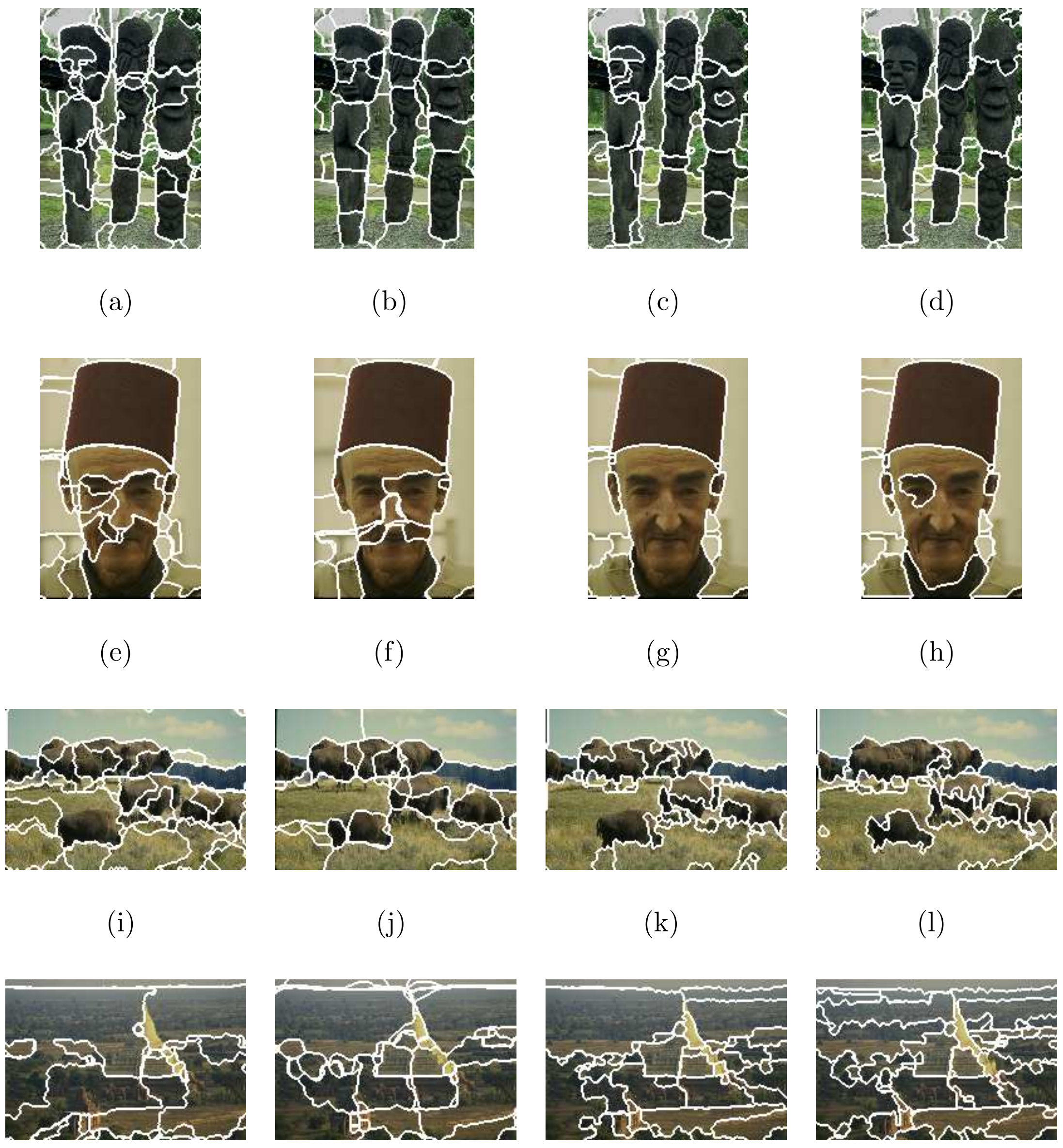


Figure 3.20: Curve evolution results on the test set. First column: GAC, second column: GVF, third column: EF′ and forth column: EF2′

is discussed in Section 2.2.3. Perona-Malik flow can be formulated as:

$$I_t = div(g\nabla I) \tag{3.18}$$

with I_0 being the original image. g is the edge stopping function, which allows edges below certain strength to be smoothed and stronger edges to be sharpened.

Another interesting work on anisotropic diffusion is self-snakes proposed by Sapiro [50]. This work uses same partial differential equations as geodesic active contours (except the constant expansion term) but instead of evolving a curve, all level sets of the image are evolved together as if they each are curves. Self-snakes is formulated as a linear combination of a smoothing term and a sharpening term. The smoothing occurs in the tangential direction of the level sets of the image, and sharpening is applied in the perpendicular direction. The PDEs can be written as:

$$I_t = \alpha g \mathrm{div}\left(\frac{\nabla I}{\|\nabla I\|}\right)\|\nabla I\| + \beta \nabla g \cdot \nabla I \tag{3.19}$$

where α and β are constant weighting factors. In this equation, g is defined as in the case of geodesic active contours and the curvature of the level sets is $\kappa = div\left(\frac{\nabla I}{\|\nabla I\|}\right)$. The first term is the smoothing term and smooths the level sets of the image proportional to their curvature. The second term is used for sharpening or deblurring. Perona-Malik flow can also be decomposed into a combination of smoothing and sharpening terms. See Section 2.2.3 for more details.

3.7 Edgeflow-driven Anisotropic Diffusion

In this section we propose an anisotropic diffusion framework that utilizes Edgeflow vector field for sharpening and the associated edge function V for selective smoothing. Similar to the edgeflow-driven curve evolution framework, we define our anisotropic diffusion framework as:

$$I_t = \alpha V \kappa \|\nabla I\| + \beta \vec{S} \cdot \nabla I \tag{3.20}$$

Our objective is to compare Edgeflow-driven anisotropic diffusion framework to Perona-Malik flow and self-snakes. First of all it is easy to see the similarities between self-snakes and (3.20). Both of them are based on combining smoothing and sharpening terms and $\gamma = \frac{\beta}{\alpha}$ defines the tradeoff between sharpening and smoothing. Basically, increasing γ increases the number of regions and detail in the final segmentation.

Two extreme cases for both self-snakes and Edgeflow-driven diffusion are 1) $\alpha = 0$ with $\beta > 0$, and 2) $\beta = 0$ with $\alpha > 0$. If β is set to zero in self-snakes, then the boundary will be smoothed by a curvature based flow. This is demonstrated in the first two rows of Fig. 3.21. Most boundaries are completely displaced and lost. This is the expected behavior of mean curvature smoothing. On the other hand, if alpha is set to zero, we do not expect the sharpening term to displace the boundary. Contrary to our expectations, applying self-snakes even with α set to zero ($\sigma = 1.5$), displaces the boundary and smooths it as shown in Fig 3.21, third and fourth rows (maybe more correctly, only smooth boundaries are preserved and sharpened). Edgeflow-driven flow does not have the same behavior

(Fig. 3.22). In these examples, g in self-snakes and $\vec{S}$ and V in Edgeflow-driven flow are updated after each iteration. The results show that self-snakes displace the edges. This effect of displacement is more severe with increasing scale.

3.8 Evaluation of Anisotropic Diffusion

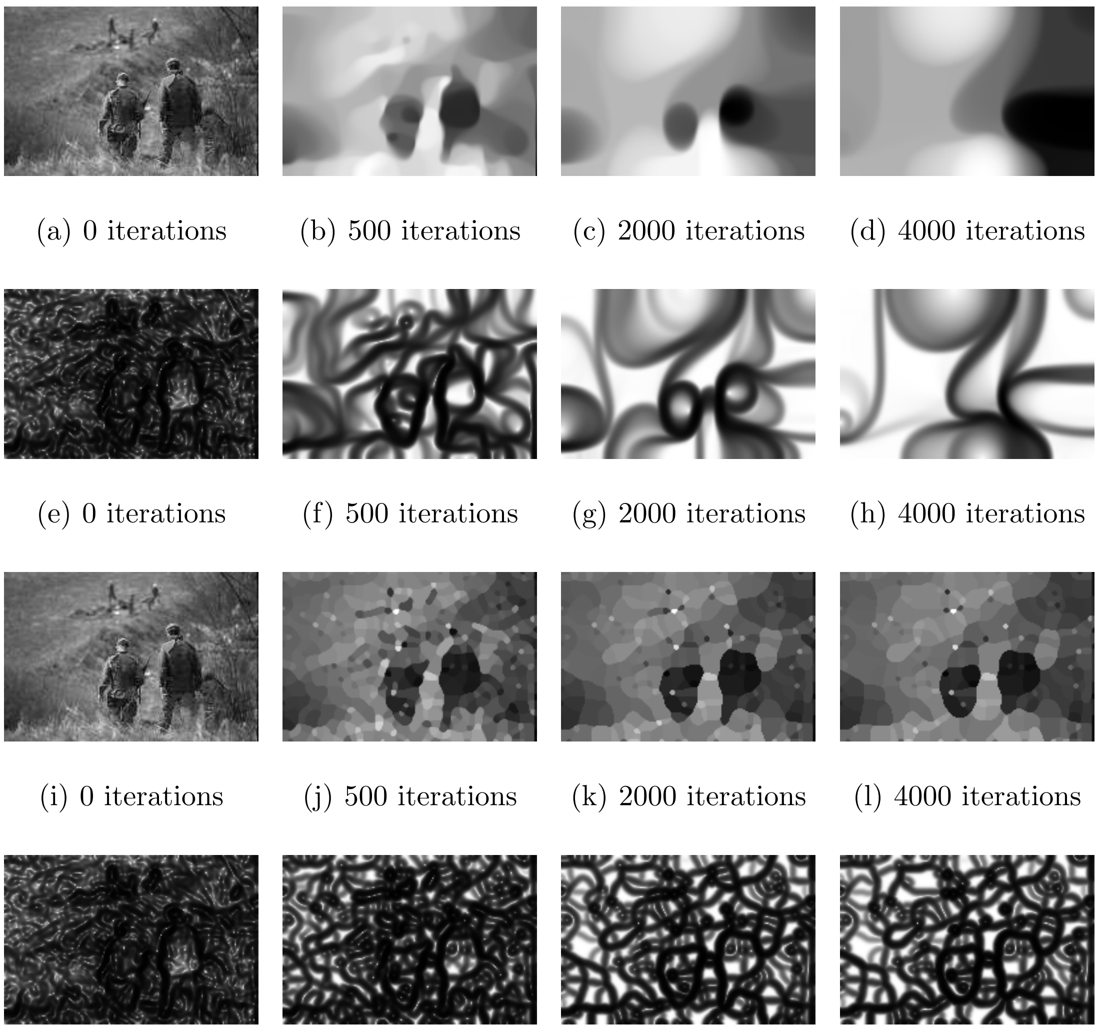
We evaluate and compare three anisotropic diffusion techniques namely, Perona-Malik Flow (PM), self-snakes (SS), and Edgeflow-driven flow (EFD or EF2D depending on the Gaussian offset). EFD and EF2D use same Edgeflow vector fields as EF$'$ and EF2$'$ as defined in Section 3.5.3. The evaluation framework in principle follows the framework introduced in Section 3.4. The training and test sets are the same. Only difference is that instead of reducing the size to 192×128, we resize the images to 240×160 for anisotropic diffusion. We need to run each of these diffusion techniques until their convergence. Instead of using a heuristic method to check convergence, we decided to let each diffusion to run 5000 iterations, which is almost always enough for these diffusion techniques to converge.

3.8.1 Training Perona-Malik Flow (PM)

Matlab source code for PM has been provided by its authors in [42, Chapter 3]. This code uses exponential nonlinearity for the edge stopping function:

$$g(\|\nabla I\|) = e^{-(\|\nabla I\|/K)^2} \tag{3.21}$$

Since this is the implementation favored by its authors, we will directly follow this code in our experiments. For completeness we reproduce the source code in



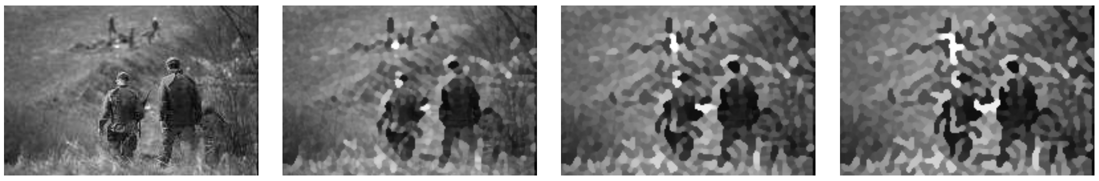
(a) 0 iterations (b) 500 iterations (c) 2000 iterations (d) 4000 iterations

(e) 0 iterations (f) 500 iterations (g) 2000 iterations (h) 4000 iterations

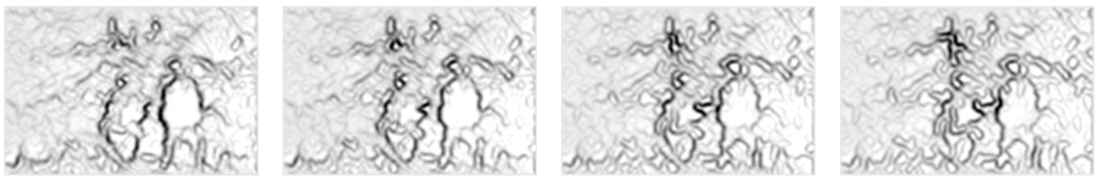
(i) 0 iterations (j) 500 iterations (k) 2000 iterations (l) 4000 iterations

(m) 0 iterations (n) 500 iterations (o) 2000 iterations (p) 4000 iterations

Figure 3.21: Self-snakes image diffusion. First two rows correspond to the diffusion and edge function g for the case when β is set to zero. Third and forth rows correspond to the diffusion when α is set to zero.



(a) 0 iterations (b) 500 iterations (c) 2000 iterations (d) 4000 iterations



(e) 0 iterations (f) 500 iterations (g) 2000 iterations (h) 4000 iterations

Figure 3.22: Edgeflow-driven image diffusion for $\alpha = 0$. First row shows the diffusion process and the corresponding edge functions V are given in the second row.

Algorithm 3.1.

The parameter K works like a threshold for edge strength. For small K, more edges are preserved and for large K only strong edges are preserved. In our experiments we set K from 1 to 10 with increments of 1 and estimate the optimum value for K using precision, recall and F-measure as described in the curve evolution section. The range of K is chosen such that for $K = 1$, unreasonably large number of pixels of the image are labeled as the edges and for $K = 10$ most of the edges of the image are missed. There is no scale to adjust in PM so K is the only parameter to tune.

Perona-Malik flow was originally demonstrated for edge detection. On the other hand, when anisotropic diffusion converges, we can actually obtain a proper segmentation for which the contours are closed. To generate the segmentation

Algorithm 3.1 Source code for Perona-Malik flow with exponential nonlinearity. Reproduced from [42, Chapter 3].

```
function [outimage] = anisodiff(inimage,iterations,K)
lambda=0.25;   outimage = inimage;   [m,n] = size(inimage);
rowC = [1:m];     rowN = [1 1:m-1];    rowS = [2:m m];
colC = [1:n];     colE = [1 1:n-1];    colW = [2:n m];
for i=1:iterations,
  deltaN = outimage(rowN,colC) - outimage(rowC,colC);
  deltaE = outimage(rowC,colE) - outimage(rowC,colC);
  fluxN = deltaN .* exp( - (1/K) * abs(deltaN) );
  fluxE = deltaE .* exp( - (1/K) * abs(deltaE) );
  outimage = outimage + lambda *
    (fluxN - fluxN(rowS,colC) + fluxE - fluxE(rowC,colW));
end;
```

boundary from the diffused images, we propose the following:

- First let the anisotropic diffusion run for 5000 iterations. At the end, this diffusion results in a piecewise homogenous image $I_d(x, y)$.
- To extract the boundaries from I_d, calculate the gradient magnitudes $f(x, y) = \|\nabla I_d\|$. To identify boundary and non-boundary points, f needs to be thresholded. If the histogram of f is visualized, it can be seen that there are two peaks, one around 0, which corresponds to non-boundary points, and the second group is around c, where $c > K$. This group corresponds to boundary points. The ideal threshold turns out to be the point right after the end of the first lump. Anything smaller causes large chunks of homogenous areas to be labeled as edges and any higher threshold causes legitimate edge points to be missed. This threshold we described is equal to K. This

is expected since Perona-Malik flow sharpens edges stronger than K while smoothing out other edges.

- Threshold f at K. This results in a binary segmentation $B_1(x, y)$, but at this point the boundaries are not as thin as we want them. In our evaluation framework, edge localization is very important.

- Apply standard non-maxima suppression to f using the gradient orientations and then threshold it at K. This results in another binary image $B_2(x, y)$. The boundaries are thinned as we wanted but they are not always closed anymore.

- From B_1, find and label connected non-boundary regions. Let us call these regions R_i where $i = 1$ to N. We would like to grow these regions until the boundaries are thinned and are 8-connected.

- Identify the set of boundary points b_1 from B_1 and b_2 from B_2. For each R_i, identify the points that are on the boundary of R_i. Grow each of R_i outwards iteratively. The rules for region growing are as follows.

  – First grow the regions only towards the boundary points that are in the set $b_3 = b_1 \setminus b_2$. It is ideal for the final boundaries to include the non-maxima suppressed edges, b_2, as much as possible. Identify combined set of points that are on the boundary of all R_i and that are an element of b_3 and that are not 4-neighbors of more than one region. Sort these points based on their gradient strength $f(x, y)$. Grow regions starting with the point that has the smallest f value. For each pixel, check to see if it still neighbors with only one region. If not, don't grow the region to this pixel. After the first iteration of region growing, find the new boundary points of R_i and continue growing until convergence.

  – Repeat the region growing step described in the previous step but take $b_3 = b_1$.

- After the convergence of the last step, we obtain a proper segmentation with thinned boundaries as desired. This process can be easily and very efficiently implemented.

A flowchart for this algorithm is presented in Fig. 3.23.

Experiments on the training set of 20 images show that the optimum value of K is 3.67 with a performance of 0.551. The general observation from these experiments is that PM's biggest problem is the poor handling of noise and clutter. Since there is no explicit smoothing, much of the noise is labeled as 1 pixel sized segments, while some legitimate boundaries are eliminated.

3.8.2 Training Self-Snakes (SS)

As we discussed in Section 3.7, SS has a problem with displaced and smoothed boundaries. For scales larger than $\sigma = 1$, the boundaries are completely out of sync with the image. Based on this perspective, we only evaluate SS at scales $\sigma =$ 0.25,0.50,0.75,1. The scale corresponds to the Gaussian smoothing used in calculating g. At each scale, we vary γ to find the optimal point. For values of

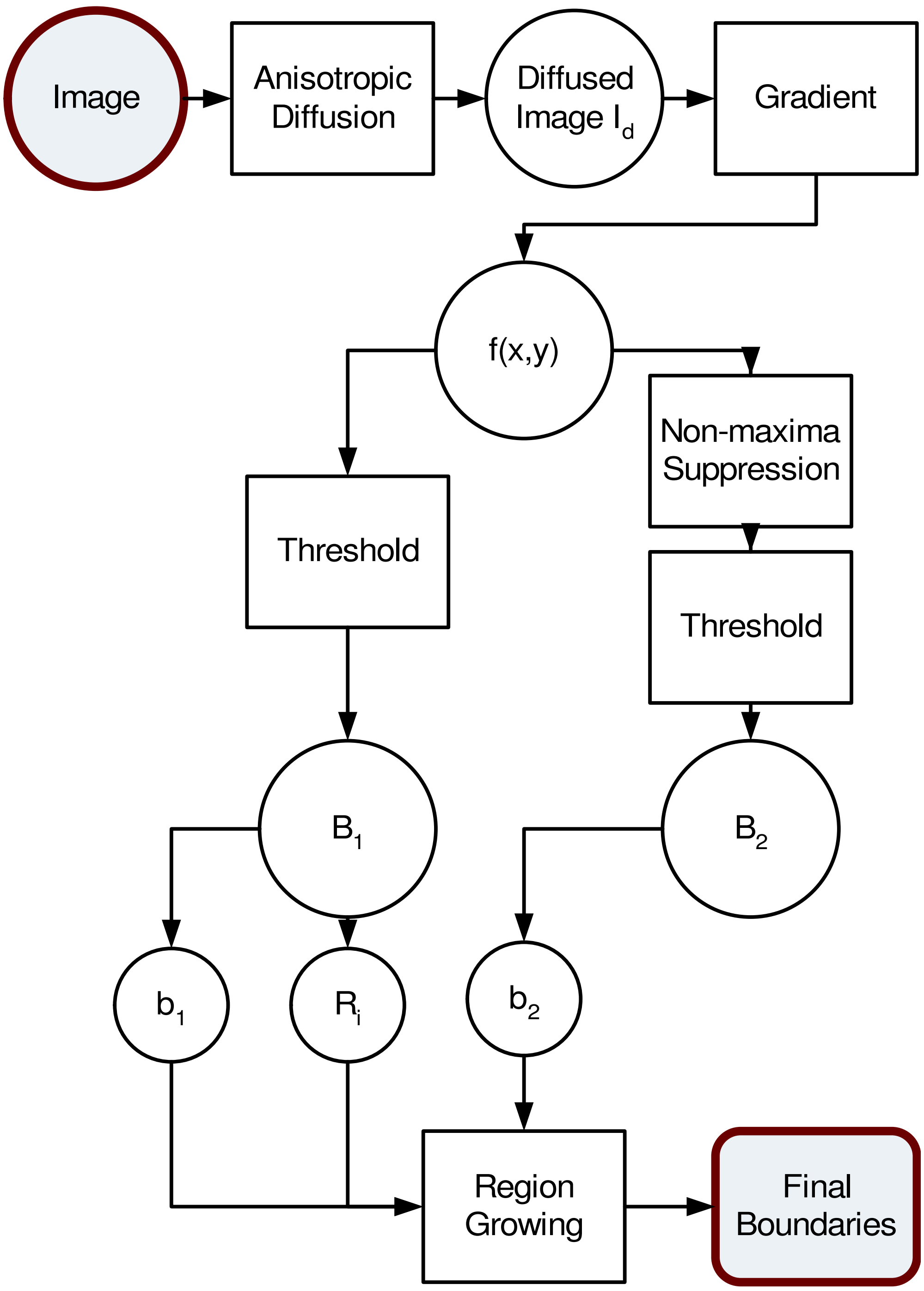
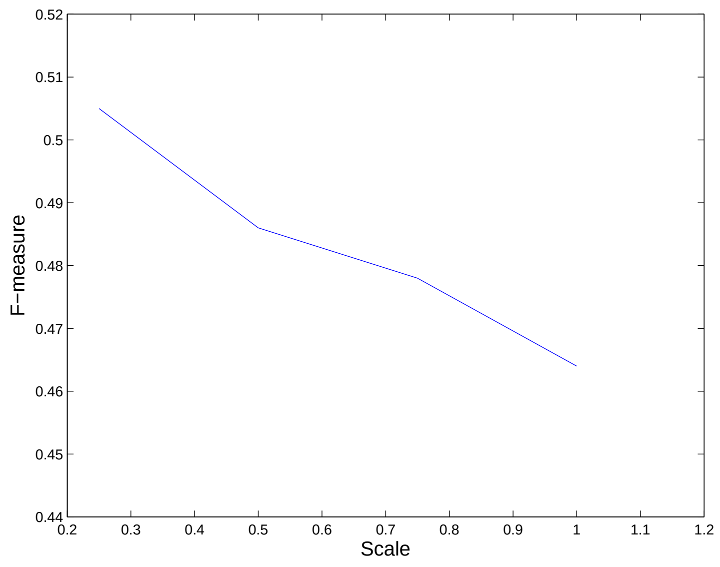


Figure 3.23: Flowchart of the algorithm for capturing thin boundaries from piecewise homogenous diffused image, generated by anisotropic diffusion.



(a)

Figure 3.24: Performance of self-snakes on the training set.

γ = 0.5,1,3,5,7,10,15 precision, recall and the F-measure are calculated and the optimum value for γ is found by linearly interpolating in between these values. During the experiments we check to see if our initial sampling of γ introduces any ambiguity and if so we extend our experiments for a denser sampling of γ.

The segmentation boundaries are generated same as PM. The threshold is set to 1. The performance with increasing scale is plotted in Fig. 3.24. As expected, the performance drops quickly with increasing scale. The optimum scale is 0.25 with the performance of $F = 0.505$ for $\gamma = 0.97$.

3.8.3 Training Edgeflow

We investigated various design decisions regarding Edgeflow in Section 3.4 when we evaluated curve evolution. We found that EF$'$ and EF2$'$ gave the best performance. We define EFD and EF2D for anisotropic diffusion based on the same vector field $\vec{S}$ and edge function V as in EF$'$ and EF2$'$.

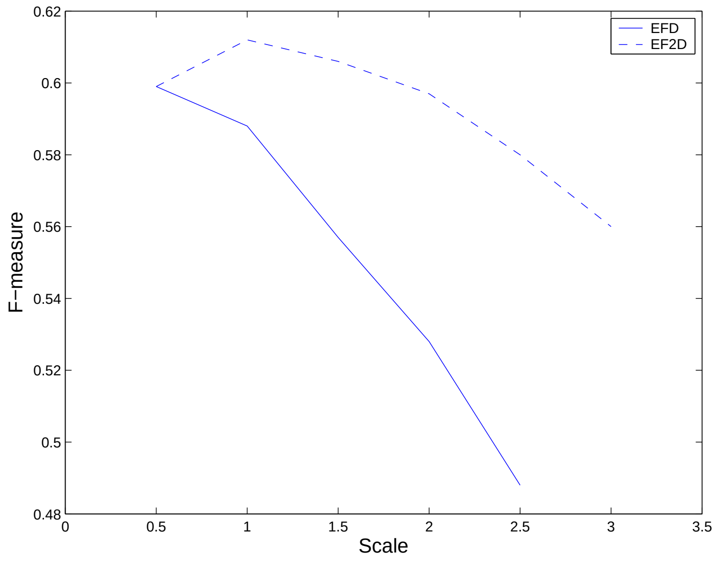
During the diffusion process, ideally, Edgeflow vector field and the edge function V are regenerated after each iteration. Unfortunately this introduces extra computational burden. Instead we choose to calculate Edgeflow only once at the beginning and reuse it. In our experiments, not updating Edgeflow iteratively did not cause performance degradation. After applying (3.20) to an image, a piecewise homogenous image, I_d is obtained. To further enhance the edges, we also apply PM with K=1 to I_d right before we generate the boundaries. Since K is chosen very low, all the regions captured by (3.20) are preserved.

Fig. 3.25 shows the relative performance of EFD versus EF2D for varying scales. EF2D performs better and peaks at $\sigma = 1$ for $\gamma = 1.0$ ($F = 0.612$). EFD's peak performance is at $\sigma = 0.5$ for $\gamma = 1.15$ ($F = 0.599$).

3.8.4 Test results

A summary of performance results on the test set of 100 images is given in Table 3.3. EFD and EF2D clearly outperform both SS and PM. These results are also much better than the results we obtained using curve evolution.

Fig 3.26 shows segmentation results for PM, SS, EFD and EF2D on four test images. Figures 3.27 and 3.28 present more segmentation results for EF2D on the



(a)

Figure 3.25: Performance comparison of EFD versus EF2D on the training set.

Technique	σ	F-measure on training set	F-measure on the test set
PM	NA	0.551	0.513
SS	0.25	0.505	0.476
EFD	0.5	0.599	0.569
EF2D	1	0.612	0.565

Table 3.3: Summary of the optimum values for anisotropic diffusion

test set.

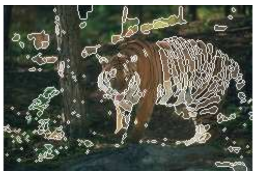
3.9 Conclusion and Discussion

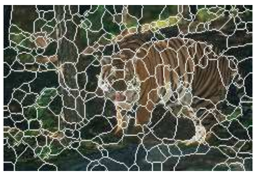
In this chapter we designed and evaluated two types of variational segmentation methods, namely curve evolution and anisotropic diffusion. For the general segmentation setup, our anisotropic diffusion shows much better performance compared with the curve evolution framework. While it can be used for complete segmentation of an image, curve evolution is best suited for extracting a single boundary at a specific part of the image. Manual and interactive segmentations especially can benefit by using curve evolution. Even for semi-automated segmentation, our findings in this chapter hold. The optimum parameters we discovered for curve evolution can be used when semi-automated detection of natural image boundaries are desired.

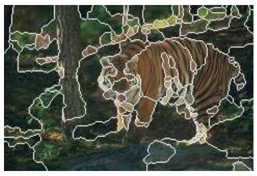
We proposed new curve evolution and anisotropic diffusion techniques both based on the Edgeflow vector field. Our evaluations show that by utilizing Edgeflow-based variational methods, we are able to outperform state of the art and well known variational techniques on natural images.

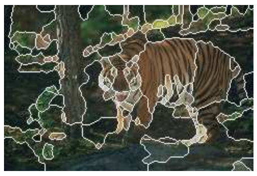
The evaluations are done using gray scale intensities. It is clear that utilizing color and texture cues should improve the performance. We leave this as future work, but we expect that Edgeflow-based methods would outperform other variational method considering that Edgeflow's original target has been texture features and texture-rich images.

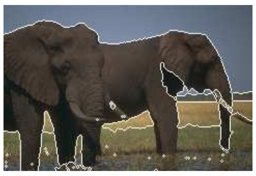
We have compared our curve evolution technique to gradient vector flow (GVF)

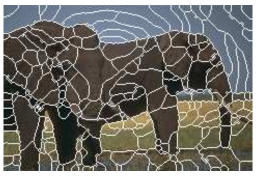

(a)

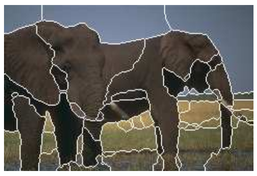

(b)

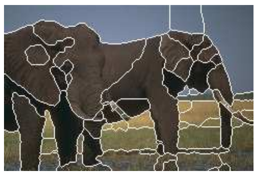

(c)

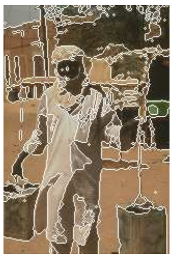

(d)

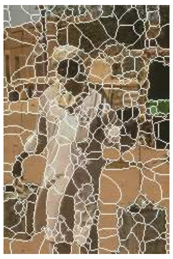

(e)

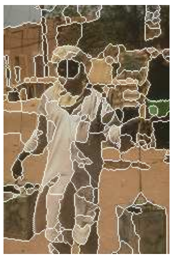

(f)

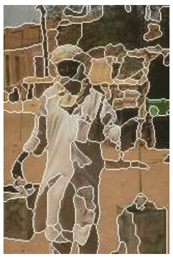

(g)

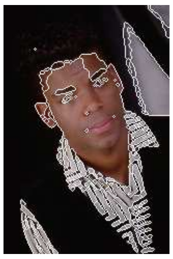

(h)

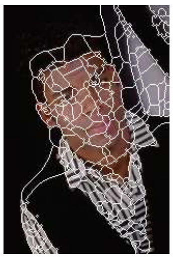

(i)

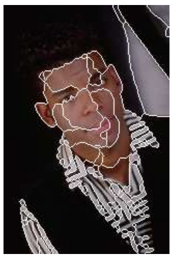

(j)


(k)


(l)


(m)


(n)


(o)

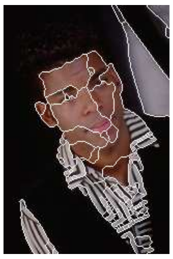
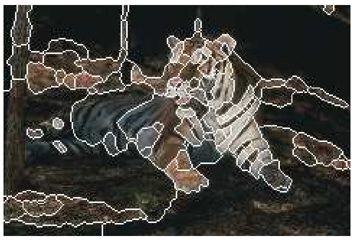
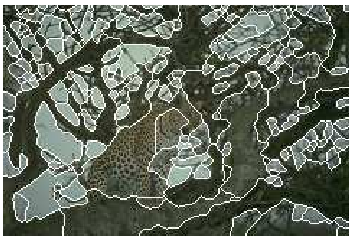
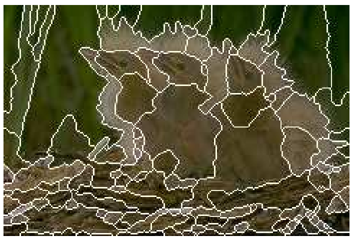
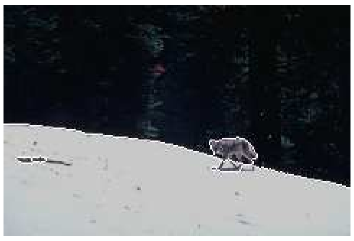

(p)

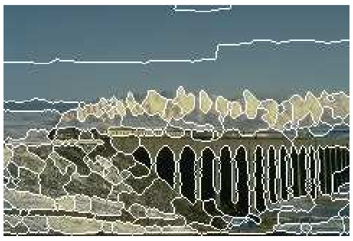
Figure 3.26: Anisotropic diffusion results on the test set. First column: PM, second column: SS, third column: EFD and forth column: EF2D

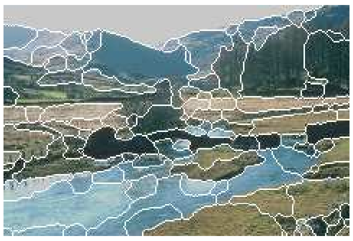

(a)

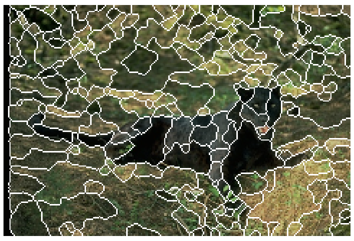

(b)

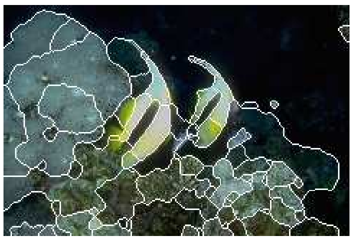

(c)

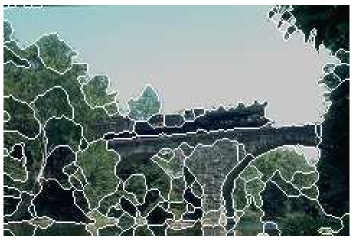

(d)

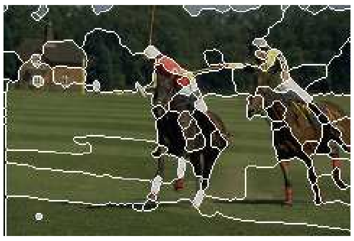

(e)

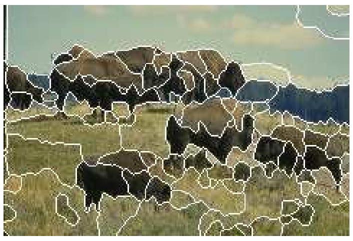

(f)


(g)


(h)


(i)


(j)


(k)

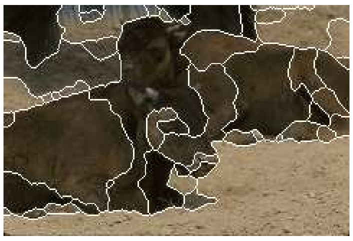
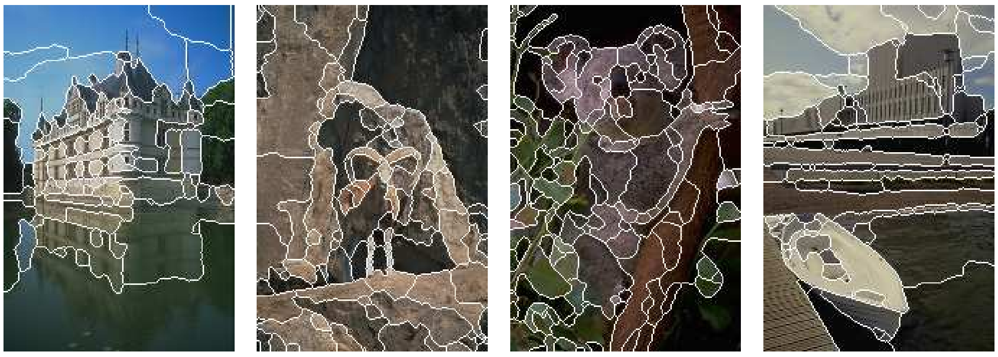
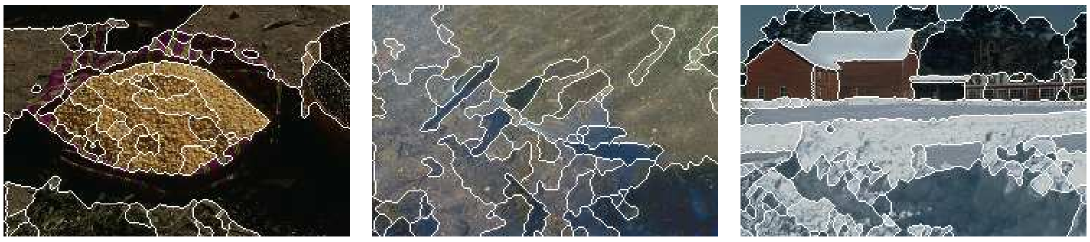

(l)

Figure 3.27: EF2D results on the test set.



(a) (b) (c) (d)



(e) (f) (g)

Figure 3.28: EF2D results on the test set.

and geodesic active contours (GAC). While GAC showed respectable performance, GVF was very disappointing on natural images.

For our experiments regarding anisotropic diffusion, self-snakes, which is an extension of GAC for anisotropic diffusion, did not perform well. Compared to the original anisotropic diffusion, our Edgeflow-based technique performed significantly better both on the training and the test sets. Edgeflow-driven anisotropic diffusion is probably the most promising variational method among the ones we tested in this chapter. Perona-Malik flow (PM) captured certain boundaries very accurately while missing some other obvious ones. Also because of the high noise sensitivity of PM, some of the noise areas are labeled as regions.

We utilized certain simplifications mostly driven by computational concerns. The images are resized to 192×128 for curve evolution and 240×160 for anisotropic diffusion. Our curve instantiation framework, manual tuning of the curvature-based term and convergence criteria, might be suboptimal. On the other hand, we evaluated all techniques under the same conditions without bias.

There are obviously some open questions with our evaluation framework. We have taken recall and precision of the edges as equally important. The main reason for this decision was consistency with previous work on Berkeley segmentation data set. In reality, the relative importance of recall and precision are very application specific. For the case of general segmentation, it is our opinion that recall is more important than precision. It is easier to merge regions through post-processing but if a boundary is missed initially, it is very difficult to recover it later on. Also in our experiments, all edges are weighted as equally important.

In reality this is not strictly true. For example, for the task of object recognition, some boundaries might be critical while some others can be safely ignored. Moreover, some types of edges such as corners and junctions contain more semantic information about the objects and should have been weighted more.

Another limitation with our framework is the size of the training set. It is possible to estimate the optimum parameters more accurately if the training set is larger. In one sense, we are trying to model natural images, which would require at least thousands, maybe much more, images in the training set. Unfortunately with our current computational resources, it is not practicle at this point. On the other hand, the training set was diverse and the parameters we estimated would be useful for practical applications.

One of the main advantages our work is that we estimated specific parameters that can be easily utilized in practical segmentation applications. On the other hand, other previous work on Berkeley segmentation data set (mostly edge detection evaluation) do not fix all the parameters using training set. For example, in [24], all other parameters except edge detection threshold are estimated from the training set. The threshold (which is one of the key parameters) is then re-optimized on the test set and the performances are provided accordingly. This type of evaluation introduces bias towards the data set and is suspect to overlearning. Moreover, the authors demonstrated edge detection results for individual images by optimizing the threshold on each image separately. Since a fixed threshold is not estimated for the algorithm, this kind of setup can not be generalized to practical applications where the test images do not come with ground truth. On the contrary, we have optimized and fixed every single parameter of each technique using the training set.

Chapter 4

Multi-scale Boundary Detection and Color Image Segmentation

Most edge detection algorithms specify a spatial scale at which the edges are detected. Typically, edge detectors utilize local operators and the effective area of these local operators define this spatial scale. The spatial scale usually corresponds to the level of smoothing of the image, for example, the variance of the Gaussian smoothing. At small scales corresponding to finer image details, edge detectors find intensity jumps in small neighborhoods. At the small scale, some of these edge responses originate from noise or clutter within the image and these edges are clearly not desirable. More interesting edges are the ones that also exist at larger scales corresponding to coarser image details. When the scale is increased, most noise and clutter is eliminated in the detected edges, but as a side effect the edges at large scales are not as well localized as the edges at smaller scales. For example, it has been shown [5] that smoothing the image with Gaussian filters

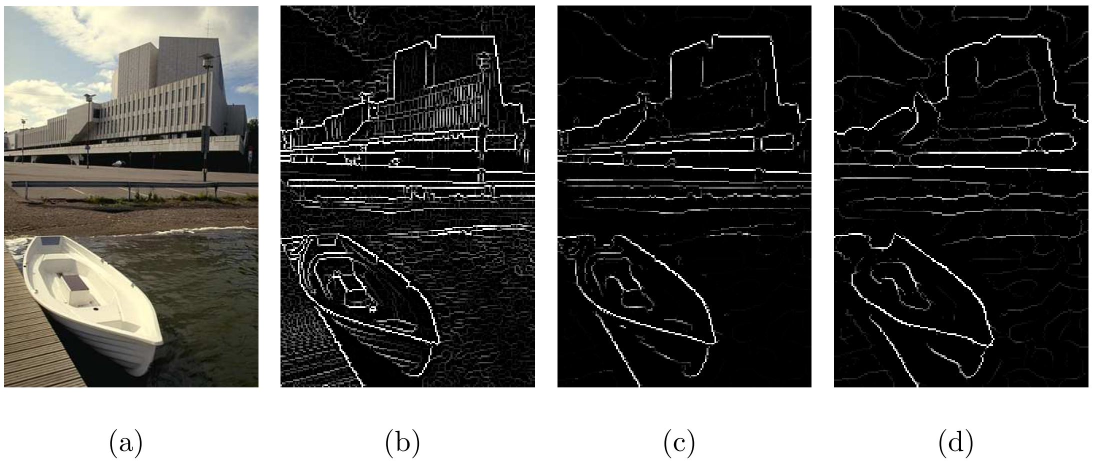


Figure 4.1: Demonstration of good localization of multi-scale edge detection. Results show that multi-scale edge detection favors edges that exist at both fine and coarse scales and edges are localized at the finer scale. a) Original image. b) Edge strengths at spatial scale $\sigma = 1$ pixel. Some clutter in the background is recognized as edges. c) Multi-scale edge detection using scales from $\sigma = 1$ to $\sigma = 4$. d) Edge strengths at scale $\sigma = 4$. Edges are not well localized.

of increasing variances causes the edges to move from their actual locations. To achieve good localization and good detection of edges, a multi-scale approach is needed. Fig. 4.1 shows an example of how multi-scale edge detection, using the methods developed in this chapter, can precisely localize edges while removing the unwanted noise and clutter. As can be seen, the multi-scale edge detection result in Fig. 4.1(c) is cleaner than the one in 4.1(b) and localizes edges better than the result shown in 4.1(d).

Edge detection and analysis of edges at multiple scales has a rich history since the early days of edge detection [104, 15, 55]. Both fine to coarse [15] and coarse to fine [105] approaches for combining edges from a range of scales have been investigated. Most of these works are based on first finding an edge representation at each scale and then combining them using certain heuristics. For fine to coarse methods, usually the smallest scale that results in a good edge detection is selected at each local neighborhood of the image. Along these lines, Canny [55] proposed a fine to coarse method called feature synthesis. First, edges are detected at small scales. Then filtering responses at larger scales are synthesized from the fine scale edges. These synthesized outputs are compared to the real coarse scale responses. Additional edges are marked only if the large scale operator output is significantly greater than the synthesized response.

A general trend in many of the multi-scale methods is combining single scale edge detector outputs at multiple scales and generating a synthesis of these edges. On the other hand, it is desirable that the multi-scale information is integrated to the edge detection at an earlier stage and the edge detection operation automatically results in multi-scale edges.

More recent multi-scale edge detection techniques are based on estimating optimum scales for local neighborhoods within the image [106-108, 53]. Lindeberg [106] analyzes the scale space representation of edge strengths $\xi(x, y, t)$, where t correspond to a continuous scale, from a differential geometric point of view. The optimum scale at a point is chosen as the scale at which ξ has a maximum in the t direction. A more interesting approach comes from Tabb and Ahuja [53]. This technique is based on designing a vector field for edge detection and image segmentation. The edges are marked as the location where the vectors diverge from each other in opposite directions. The idea of designing a vector field for edge detection is very similar to the Edgeflow technique [1]. Tabb and Ahuja create these vectors by analyzing a neighborhood around each pixel. Within a neighborhood around a pixel $p(x, y)$, vectorial intensity difference $\|I(p) - I(p')\|\overrightarrow{(p - p')}$ to its neighboring pixels p' are calculated and summed to generate the vector at $p(x, y)$. For a multi-scale representation, the optimum neighborhood size changes from pixel to pixel and needs to be estimated. The technique accepts a parameter that specifies a desired homogeneity level within regions. Using this homogeneity parameter, the neighborhood size and the spatial scale are estimated at each pixel adaptively.

Another approach to multi-scale edge detection and segmentation is anisotropic diffusion [6], which is discussed in detail in Sections 2.2.3 and 3.6. Anisotropic diffusion is based on preventing smoothing around the edge locations. This is equivalent of applying Gaussian smoothing with a spatially adaptive variance. A pixel within an homogenous region is smoothed with a Gaussian of large variance whereas a pixel close to an edge is smoothed at a smaller scale.

An important question still remains. Should the image be analyzed from fine scale to coarse scale or vice versa? In general, this should not matter for a well designed computerized system. Experiments [109] show that neurons in the visual cortex of Old World monkeys[1] are tuned from coarse scales to fine scales. It can be easily argued that at first sight we analyze a scene at a coarse scale and over time we start seeing the finer details. Similarly for a computer vision system, it is desirable that the edges exist at both coarse and fine scales, and the localization of these edges are decided at the finest scale. Note that a boundary at coarse

[1]Old World monkeys are a family of monkeys including baboons and macaques.

scale might consist of several boundaries at the fine scale when the detail level is increased.

In the next section we will design and propose a multi-scale edge detection method. Our technique is motivated from a geometrical point of view. Unlike previous work in this area, it is not necessary to estimate a scale locally. The objective is to detect edges that exist at both coarse and fine scales, and localize them at the finest scale.

4.1 Multi-scale Edgeflow vector field and edge detection

Wei and Manjunath introduced a methodology [1] for creating the Edgeflow vector field at a user defined spatial scale. This vector field is then used for edge detection and image segmentation. Image segmentation is generated in a ad hoc way from the edges by edge linking. We are more interested in the vector field design and edge detection parts of this work. As discussed in 2.3.1, Edgeflow vector field is designed in a way such that the vector flow is towards the boundary at either side of the boundary. For detecting the edges, first a way of vector propagation is applied to the vector field to enhance the edge locations. Edges are then labeled as the locations where vector field reverses its direction. To detect edge locations, x and y components of the vector field are checked for sign changes (along the x and y directions) and the pixel that changes from positive to negative is labeled as the edge pixel. The edge strength equals the absolute difference of the magnitudes at the transition. Our primary purpose in this section is to define a multi-scale vector field that is based on the Edgeflow technique. We will then utilize this multi-scale vector field for multi-scale edge detection. In Section 4.2, the same vector field will also be used for image segmentation within our anisotropic diffusion framework.

First let us summarize the changes we made to the vector field generation and edge detection procedures that are proposed by the original Edgeflow technique.

- As discussed in the Section 3.1, we choose the magnitudes of Edgeflow vectors in a different way than originally proposed in [1]. While the original Edgeflow method [1] uses the gradient of the smoothed image for computing the vector magnitudes, we utilize relative directional differences.

- We do not apply the vector propagation stage proposed in [1]. In our experiments, vector propagation stage, while eliminating some clutter from the image, did not change the final edge detection result significantly. Based on our preliminary tests on the Berkeley segmentation data set, edge detection performance for gray scale images seems to stay the same before and after vector propagation. Moreover, we need a dense vector field for our anisotropic diffusion scheme that we will discuss in Section 4.2. For this purpose, we directly look for direction reversals within the initial vector field to detect the edges.

- Based on our experiments in Chapter 3, we have found that if the Gaussian offset in the calculation of Edgeflow vector field is reduced to σ instead of

4σ, segmentation performance stays stable across a large range of scales. For an offset of 4σ, the performance quickly drops with increasing scale. This behavior is not suitable for a multi-scale implementation. Therefore we change the value of Gaussian offset to σ in our experiments. The original Edgeflow method [1] is designed with texture edge detection in mind. In our experiments, interestingly using a Gaussian offset of 4σ as suggested by the original Edgeflow method results superior texture discrimination compared to an offset of σ. The reason for this behavior is that texture is a feature defined in neighborhoods and gray scale or color are point features. For a point feature, using a Gaussian offset of σ shows more robust behavior since the operator in this case resembles the derivative of a Gaussian. On the other hand, using a Gaussian offset of 4σ means measuring the weighted difference of two non-overlapping neighborhoods (See Fig. 2.2(a)). This explains why gradient type operators have little success with texture edge detection.

Our main goals in designing the multi-scale edge detector are:

- Localize edges at the finer scales.
- Suppress edges that disappear quickly when scale is increased. These are mostly spurious edges that are detected at the fine scale because of noise and clutter in the image but do not form salient image structures.
- Favor edges or edge neighborhoods[2] that exist at both fine and coarse scales.

[2]For larger scales, edges are displaced from their original locations. For this reason, it makes more sense to discuss edge neighborhoods when larger scales are considered.

In generating our vector field, we will explicitly use a fine to coarse strategy. On the other hand, our multi-scale framework also conducts coarse to fine edge detection implicitly. Take s_1 as the finest (starting) scale and s_2 as the coarsest (ending) scale. We are interested in analyzing an image between scales s_1 and s_2 for finding the edges. The units for scales are in pixels. All images used in this chapter are of size 240×160. The interval $[s_1, s_2]$ is sampled with increments of Δs. In [105], Bergholm uses $\Delta s = 0.5$ such that dislocation of edges for successive scales is less than a pixel. Similarly, we will also set Δs to 0.5. The algorithm for generating the multi-scale Edgeflow vector field is given in Algorithm 4.1.

As can be seen, the vector field that is generated at scale s_1 is selectively updated using the vector fields from larger scales. The vector field update procedure can be interpreted as follows:

At a small scale, the vector field only exists on a thin line along the edges. Therefore, within the homogenous areas the vectors are of zero length. With increasing scale the reach–coverage area–of the vector field also gets thicker. First of all, we want to preserve the edges detected at the fine scale, which implies that we preserve strong vectors from the fine scale. We also would like to fill the empty areas with vectors from larger scales. The main reason for this is that some edges that do not exist at fine scales, the so called shadow, shading or blur edges, will be captured at the larger scales. For these reasons, we check if $\|\vec{S}(x,y)\| < \frac{M}{C}$, and if so, fill this pixel with a vector from a larger scale.

Note also that the proposed method favors edges that exist at multiple scales and suppress edges that only exist at finer scales. The strength of the edges are

Algorithm 4.1 Algorithm for generating a multi-scale Edgeflow vector field.

```
Let $I(x, y)$ be the image.
Let $C$ be a positive constant (e.g. 15).
Let $A$ be a positive constant corresponding to an angle (e.g. $\pi/4$).
Let $s_1$ be the smallest and $s_2$ be the largest spatial scale at which we are
interested in analyzing the image for edges.
Let $\Delta s = 0.5$ pixel be the sampling interval for the scale.
From $I$, calculate the initial Edgeflow vector field $\vec{S}$ at scale $s = s_1$.
while $s < s_2$ do
  Set $s = s + \Delta s$.
  Calculate Edgeflow vector field $\vec{T}$ at scale $s$.
  $M = Max(\|\vec{S}\|)$
  for all Pixel $(x, y)$ in $I$ do
    if $\|\vec{S}(x, y)\| < \frac{M}{C}$ then
      $\vec{S}(x, y) = \vec{T}(x, y)$
    else if The angle between $\vec{S}(x, y)$ and $\vec{T}(x, y)$ is less than $A$ then
      $\vec{S}(x, y) = \vec{S}(x, y) + \vec{T}(x, y)$
    else
      $\vec{S}(x, y)$ is kept the same.
    end if
  end for
end while
The final $\vec{S}$ gives the multi-scale Edgeflow vector field.
```

represented by the strength of the vectors at the edge location where the vector field changes its direction. If the vector directions match at multiple scales, this means that the edges exist at multiple scales. Based on this observation, we check the vector directions from larger and finer scales and if they match, we sum the vectors up to strengthen the edge.

Another possibility is that the edge is shifted from its original location at the larger scale. In that case, the vector at the pixel that is in between the original (small scale) edge and the shifted (large scale) edge will change its direction by 180 degrees. As we discussed before, we favor the edge localization at the finer scale. Therefore the new vector from the larger scale is ignored and the vector from the finer scale is preserved.

Figures 4.2 and 4.3 show the results of multi-scale edge detection with $s_1 = 1$, $s_2 = 4$, $\Delta s = 0.5$, $C = 15$, and $A = \pi/4$. Multi-scale edge detection results are compared to edge detection results at $\sigma = 1$ and $\sigma = 4$. These results show that the results corresponding to $\sigma = 1$ localizes edges very well but detect clutter and noise as edges. The results corresponding to $\sigma = 4$ include cleaner results but the edges are not well localized. On the other hand, by combining results from scales 1 to 4, we are able achieve edge detection results that both localize edges precisely (as in $\sigma = 1$) and create a cleaner edge detection (as in $\sigma = 4$).

The main advantage of our method is that there is no need to estimate local scales at each pixel. The vector field is designed to contain cues from multiple scales and the scale selection is implicit within the multi-scale vector field. Our approach is able to localize edges with no extra and external effort such as tracking

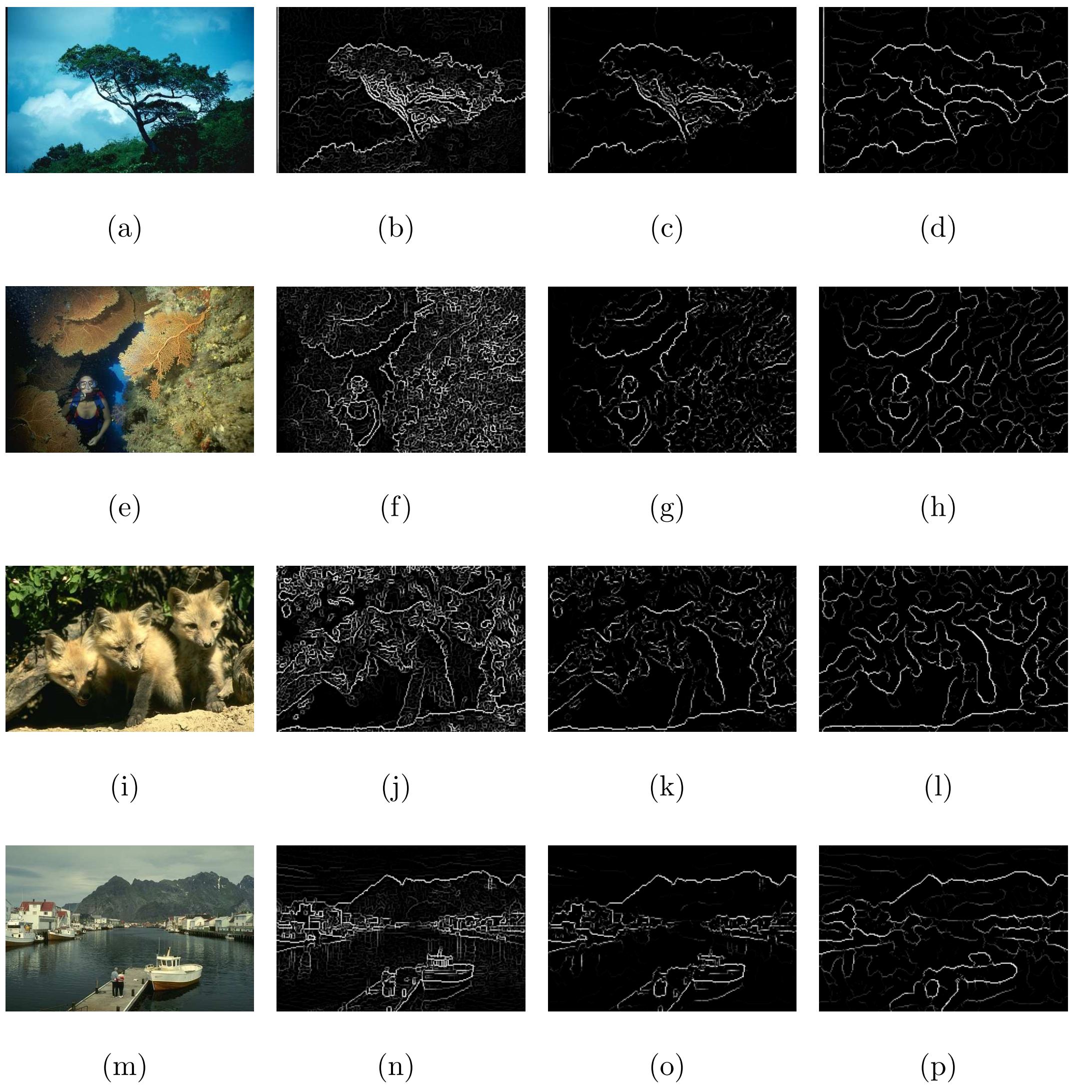


Figure 4.2: Demonstration of multi-scale edge detection. First column shows the original images. Second column shows the edge strengths at spatial scale $\sigma = 1$ pixel. Third column shows multi-scale edge detection results using scales from $\sigma = 1$ to $\sigma = 4$. Fourth column shows edge strengths at scale $\sigma = 4$ pixels.

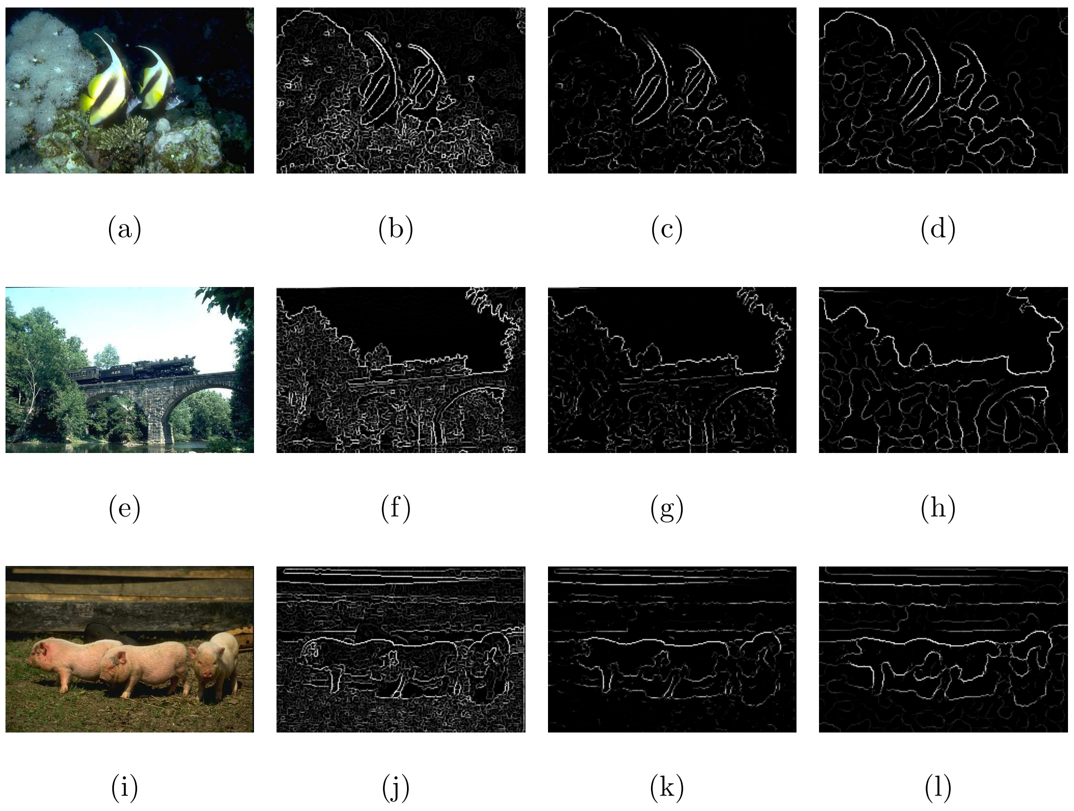


Figure 4.3: Demonstration of multi-scale edge detection. First column shows the original images. Second column shows the edge strengths at spatial scale $\sigma = 1$ pixel. Third column shows multi-scale edge detection results using scales from $\sigma = 1$ to $\sigma = 4$. Fourth column shows edge strengths at scale $\sigma = 4$ pixels.

edges or analyzing the displacement of corners, junctions etc., in scale space. Edge detection results show that the edges are localized as desired and salient structures existing at both fine and coarse scales are captured. In the next section we show a multi-scale segmentation method using the vector field designed in this section.

4.2 Multi-scale Image Segmentation

In Section 4.1, we designed a multi-scale Edgeflow vector field and utilized this vector field for multi-scale edge detection. Another interesting application of this vector field is multi-scale image segmentation. In Chapter 3, we introduced a new variational segmentation method that is based on anisotropic diffusion. This anisotropic diffusion scheme utilizes a vector field to find the boundaries. Simply replacing this vector field with the multi-scale Edgeflow vector field, we are able to achieve a multi-scale image segmentation.

The multi-scale segmentation is achieved as follows. From the multi-scale vector field, we first generate an edge stopping function V by solving a poisson equation:

$$\vec{\nabla} \cdot \vec{S} = -\Delta V \tag{4.1}$$

where $\vec{S}$ is the Edgeflow vector field. Using the vector field and edge stopping function, segmentation is defined as the convergence of the following anisotropic diffusion:

$$I_t = \alpha V \kappa \|\nabla I\| + \beta \vec{S} \cdot \nabla I \tag{4.2}$$

Fig. 4.4 demonstrates the behavior of the multi-scale segmentation compared to segmentations at the fine and coarse scales. The results show that using a multi-scale approach we are able to capture salient structures from a range of scales. It is to be noted that region merging using fine scale segmentation will not usually give the similar results as our multi-scale segmentation. For example, in Figures 4.4 (b-d), there are certain boundaries and structures that emerge as the scale increases and these boundaries are not captured or do not exist at the smaller scales. Figures 4.4 (e-h) demonstrate the excellent edge localization property of our multi-scale algorithm. Fig. 4.5 shows another example of a multi-scale segmentation and shows the better localization of the edges around the head area using multi-scale approach compared to segmentation at scale $\sigma = 6$.

Figures 4.6 and 4.7 compare segmentations at the fine scale $\sigma = 1.5$ to the multi-scale segmentations generated using $s_1 = 1.5$, $s_2 = 6$, $C = 30$ and $A = \pi/4$. These results again show the advantages of our method. The unnecessary detail that is captured at the fine scale is eliminated by the multi-scale segmentation without dislocating the edges. On the other hand, some of the legitimate boundaries are also eliminated by the multi-scale approach. These examples are intended to show the general behavior and properties of the multi-scale segmentation compared to single-scale segmentation. The results indicate that multi-scale segmentation is a promising new direction. However, much work remains in optimizing the multi-scale parameters. In contrast, the results presented in the previous chapter were optimized for the single scale method using the Berkeley segmentation data set. Optimizations and evaluations of the multi-scale segmentation is left as future work at this point.

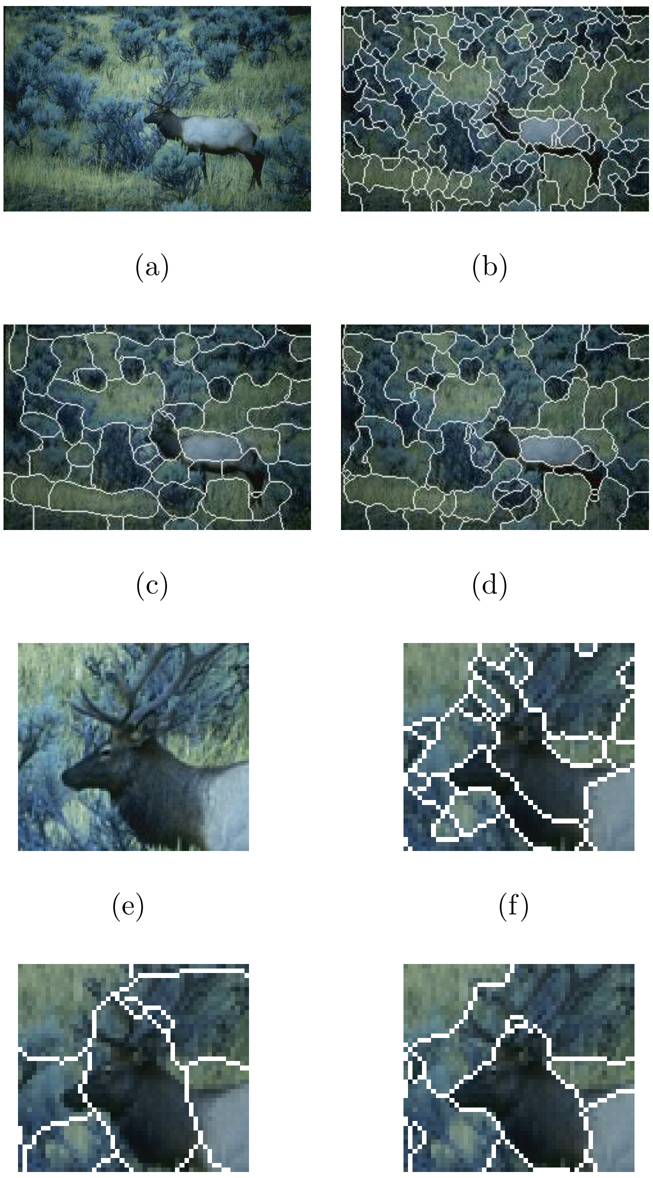


Figure 4.4: a) Original image. b) Segmentation result at spatial scale $\sigma = 1.5$. c) Segmentation result at spatial scale $\sigma = 6$. d) Multi-scale segmentations using scales $\sigma = 1.5$ to $\sigma = 6$. e) Detail around the head area. f) Detail for segmentation at $\sigma = 1.5$ g) Detail for segmentation at $\sigma = 6$ h) Detail for segmentation using scales 1.5 to 6.

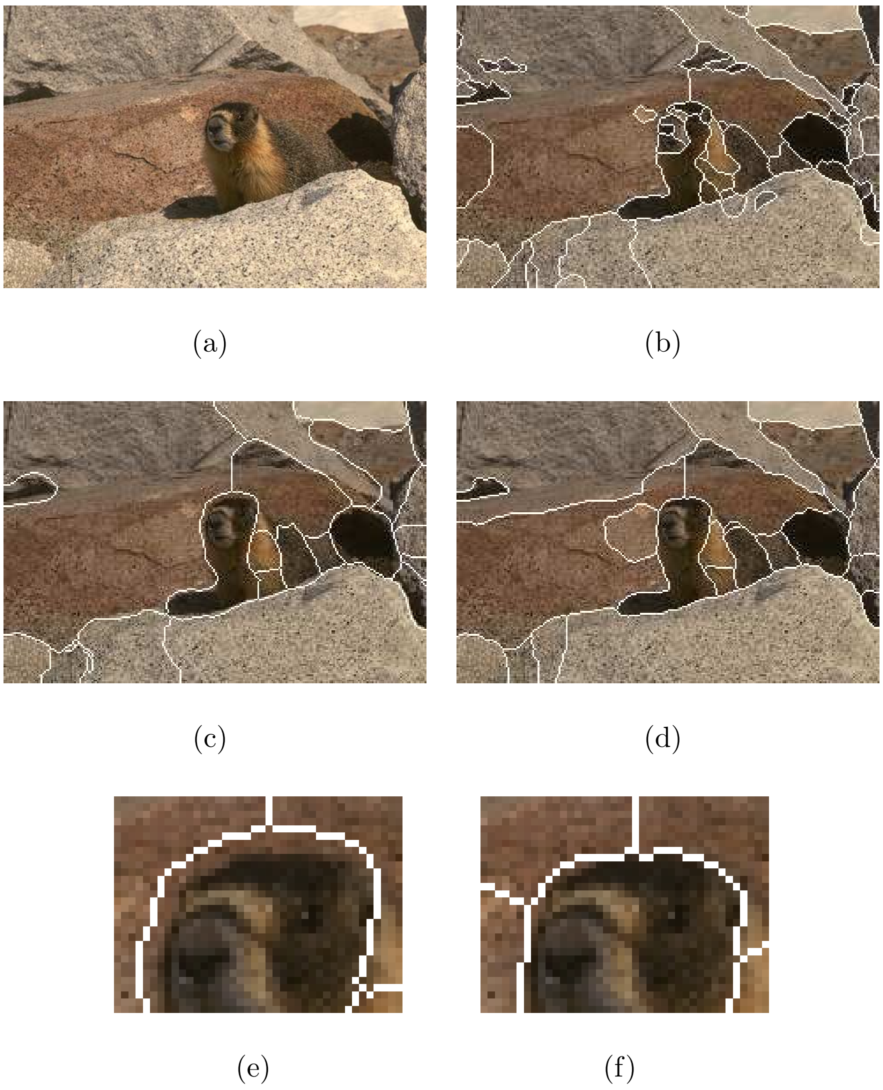
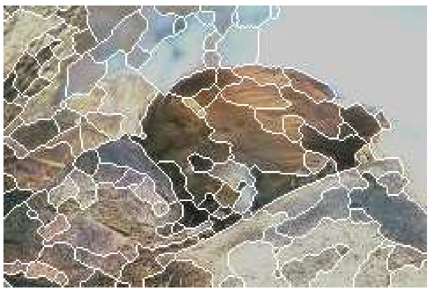
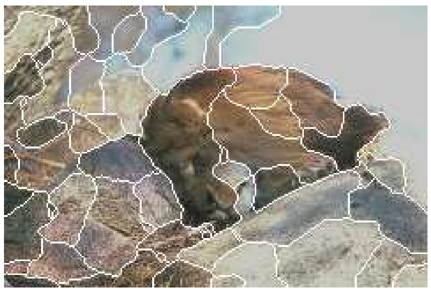


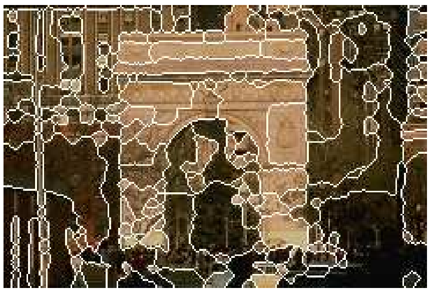
Figure 4.5: a) Original image. b) Segmentation result at spatial scale $\sigma = 1.5$. c) Segmentation result at spatial scale $\sigma = 6$. d) Multi-scale segmentations using scales $\sigma = 1.5$ to $\sigma = 6$. e) Detail around head area for $\sigma = 6$. f) Detail for multi-scale segmentation.



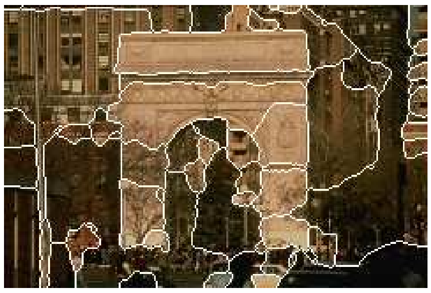
(a)



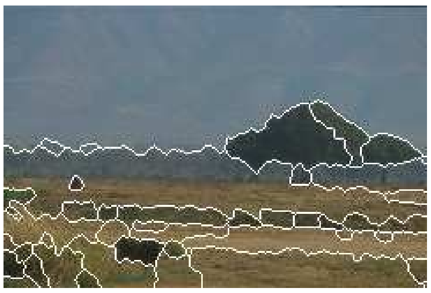
(b)



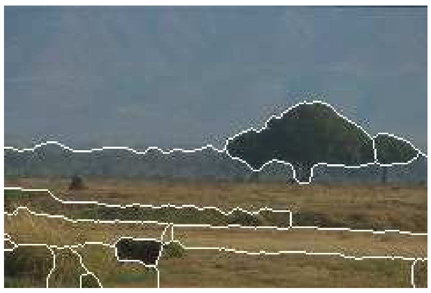
(c)



(d)



(e)



(f)

Figure 4.6: First column shows the segmentation results at spatial scale $\sigma = 1.5$. Second column shows the corresponding multi-scale segmentations using scales $\sigma = 1.5$ to $\sigma = 6$.

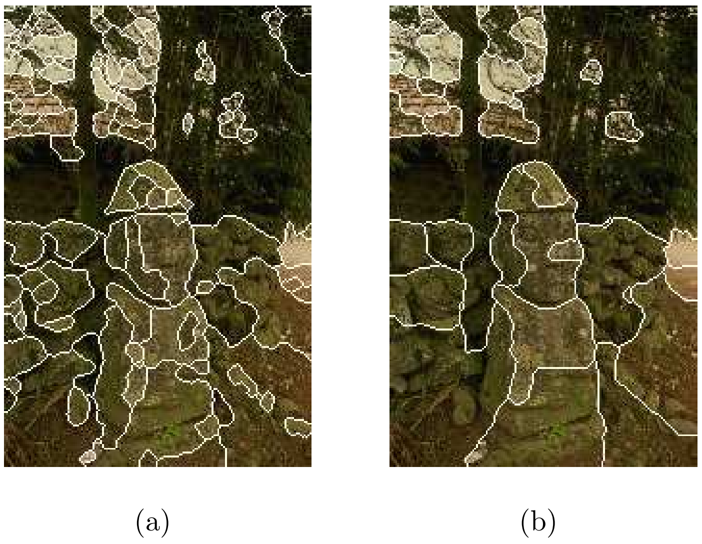


Figure 4.7: a) Segmentation at spatial scale $\sigma = 1.5$. b) Multi-scale segmentation using scales $\sigma = 1.5$ to $\sigma = 6$.

4.3 Color Image Segmentation

Our experiments in Chapter 3 and in this chapter so far are based on brightness features of the image. We perceive and understand natural images in full color and therefore our results so far (based on gray scale intensities) have been approximations of the true potential of our methods. In this section we demonstrate that full color information will make it possible to capture certain boundaries that are difficult to identify with just using the brightness component.

Almost all color images are stored or at least displayed on screen in RGB color space, which assigns three dimensional color features to each pixel. The nicety of using gray scale intensities is that the perceptual similarity can be directly measured by the absolute difference of gray scale intensities. However, the Euclidean distance between two colors might not reflect their perceptual similarity or dissimilarity. Some of the earlier methods considered each color component as separate intensity images and processed them individually while combining the results at the end. This approach is highly suboptimal and might lead to undesirable results. To reflect the perceptual color similarity of humans, several color spaces have been proposed in the literature. Among the ones we tested, CIE-L*a*b* color space [110], which is designed based on psychophysical experiments, is the most promising (we call this color space *Lab* from this point on). We also see a trend that most recent work on color edge detection and segmentation utilize *Lab* color space. The characteristics of *Lab* are explained in [4] as follows. For small distances, the perceptual similarity is related to the Euclidean distance of the color features. On the other hand, for large distances only thing we can say is that the two colors are dissimilar but the level of dissimilarity is not apparent from the Euclidean distance between these colors. Our implementation of color space conversion from RGB to *Lab* is based on the C code written by Yossi Rubner and Mark Ruzon, which is available at http://robotics.stanford.edu/ ruzon/compass/.

A common approach [45] to extending anisotropic diffusion techniques to color images is to derive the vector field, edge function, and the color gradients using all three color components, and then to apply diffusion to each color component separately. In our method, these three diffusions are coupled through the Edgeflow vector field and the edge function. For this purpose we need definitions of Edgeflow vector field and the gradient on vector valued images since color features can be thought as three dimensional vectors at each pixel.

A derivation of gradient for vector-valued images is given by Di Zenzo [44].

Let $\vec{I} : \mathbb{R}^2 \rightarrow \mathbb{R}^N$ be a vector-valued image. Define a 2×2 matrix M as:

$$M = \begin{bmatrix} \vec{I}_x \cdot \vec{I}_x & \vec{I}_x \cdot \vec{I}_y \\ \vec{I}_x \cdot \vec{I}_y & \vec{I}_y \cdot \vec{I}_y \end{bmatrix} \tag{4.3}$$

where $\vec{I}_x$ and $\vec{I}_y$ are the derivatives of $\vec{I}$ with respect to x and y. Let $\vec{v}$ be a unit vector at direction θ. The directional derivative of $\vec{I} = [I^1 \dots I^N]^T$ at the direction $\vec{v}$ is given as:

$$(\nabla_\theta \vec{I}(x,y))^2 = \vec{v}^T M \vec{v} \tag{4.4}$$

Then the gradient, highest rate of change, magnitude and direction are given by the square root of larger eigenvalue of M and the corresponding eigenvector. In our algorithm, the gradient is used for finding boundaries after diffusion converges.

To be able to conduct the anisotropic diffusion process on color images we also need a definition of Edgeflow for vector-valued images. Wei and Manjunath [1] extended Edgeflow to color and texture images by finding the prediction error for each feature component and combining them by summation. On the other hand, this approach is not well founded. We suggest a different approach by using the definition of the directional derivatives for vector-valued images (4.4). By rearranging terms, (4.4) can be rewritten as:

$$\nabla_\theta \vec{I}(x,y) = \left\| \begin{bmatrix} \vec{I}_x & \vec{I}_y \end{bmatrix} \vec{v} \right\|_2 = \left\| \begin{bmatrix} \nabla_\theta I^1 \\ \vdots \\ \nabla_\theta I^N \end{bmatrix} \right\|_2 \tag{4.5}$$

Similarly, by using the result from (4.5) the prediction error for a vector-valued image at the direction θ can be calculated as:

$$Error(\sigma, \theta) = \left\| \begin{bmatrix} Error_\theta I^1 \\ : \\ Error_\theta I^N \end{bmatrix} \right\|_2 \qquad (4.6)$$

This result shows that the prediction error for a vector-valued image is the L_2 norm of a vector created by the prediction errors of individual feature components. Original Edgeflow method has extended error calculation to vector-valued images in a similar way but using L_1 norm instead of L_2 norm, which turns out to be an approximation of (4.6). On the other hand, using L_2 norm we achieve a more accurate extension of Edgeflow vector field to vector-valued images.

Figures 4.8 and 4.9 show a comparison of segmentations corresponding to gray scale and color features. The parameters for gray scale segmentations are chosen as the optimum values that are estimated in Chapter 3 ($\sigma = 1$ and $\gamma = 1.0$). Color segmentations also use the same parameters. In each of these examples, gray scale segmentation is not able to segment the object of interest properly whereas color segmentation is successful most of the time. For example, the following are missed by the gray scale segmentation and correctly identified by the color segmentation: the back of the kangaroo in Fig. 4.8(a), various parts of the zebras in Fig. 4.8(d), the pot in Fig. 4.8 (g), upper left side of the building in Fig. 4.8(j), the roof and the mountain line in Fig. 4.9(a), the tail of the alligator in Fig. 4.9(d), the green snake in Fig. 4.9(g), the back of the tiger in Fig. 4.9(j), wings and the tail of the airplane in Fig. 4.9(m).

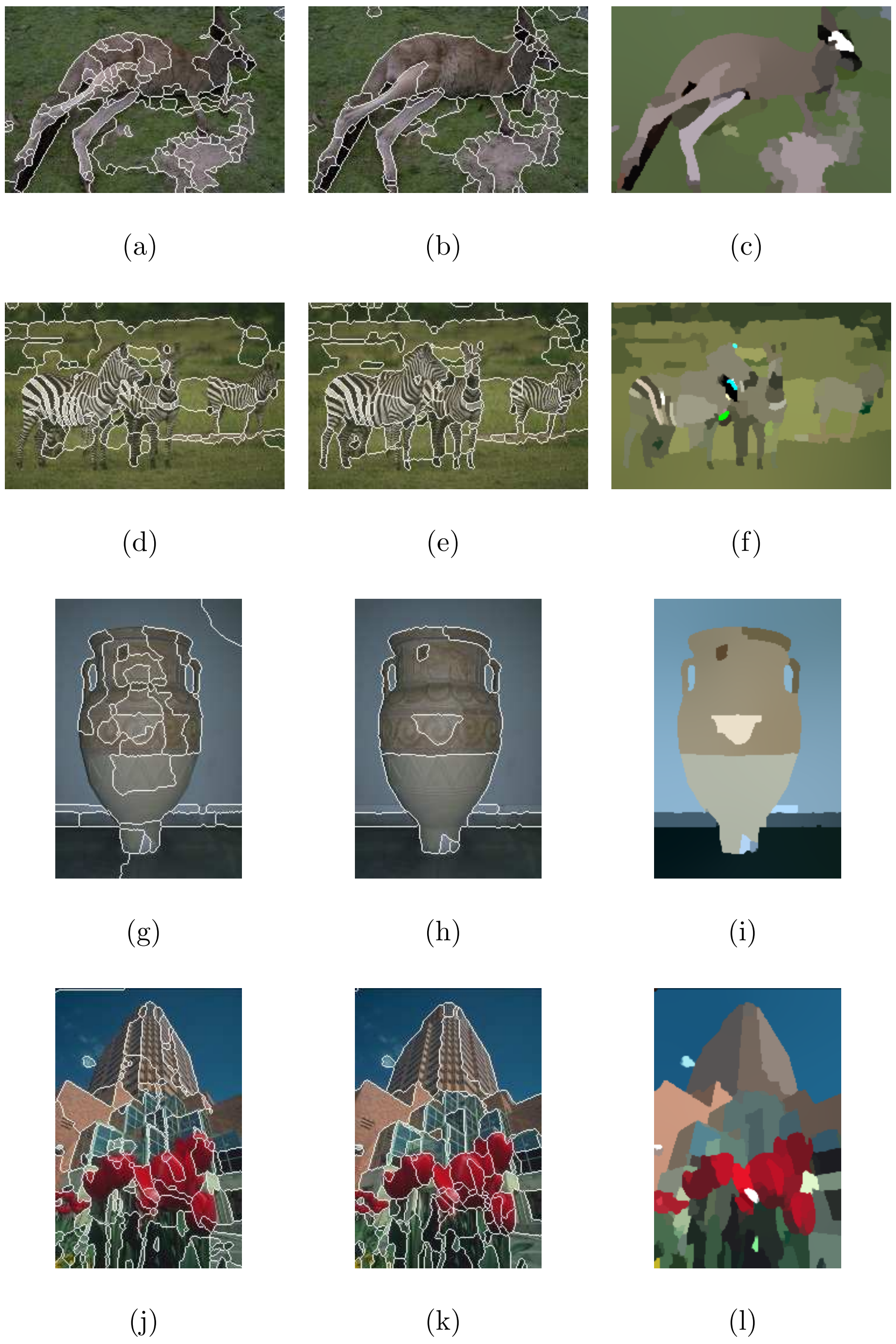


Figure 4.8: First column shows the segmentation results using gray scale features at spatial scale $\sigma = 1$. Second column shows the corresponding color segmentations. Third column shows the color anisotropic diffusion result.

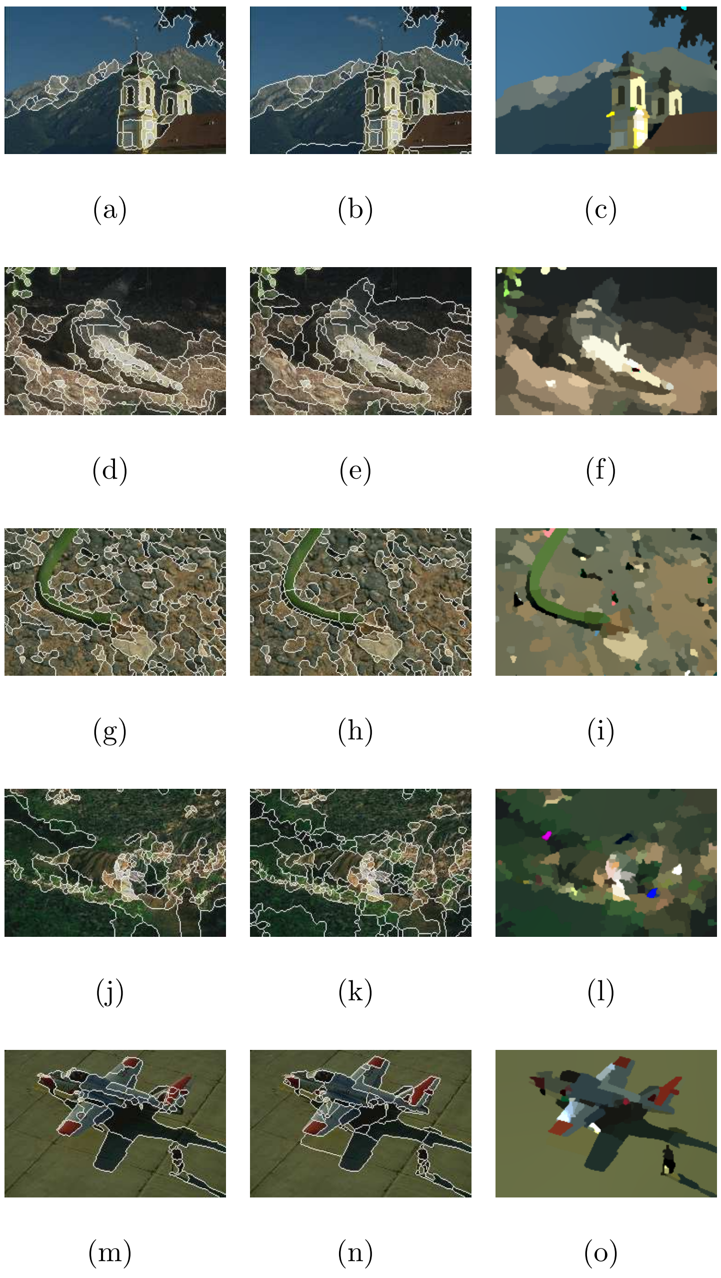


Figure 4.9: First column shows the segmentation results using gray scale features at spatial scale $\sigma = 1$. Second column shows the corresponding color segmentations. Third column shows the color anisotropic diffusion results after convergence.

4.4 Conclusions

In this chapter, we have shown that using multi-scale techniques and color information, edge detection and segmentation quality on natural images can be improved significantly. We proposed a novel multi-scale edge detection and vector field design scheme. Our approach eliminated the need for scale selection and edge tracking, which has been the main focus of the previous work in this area. Our objective is to find and favor edges that exist at a wide range of scales and localize these edges at finer scales. This work is then extended to multi-scale segmentation using our anisotropic diffusion scheme, which is introduced in Chapter 3.

Natural images are seen and perceived in color. On the other hand we have been analyzing segmentation based on the brightness features mainly for the reasons of simplicity. In this chapter we also investigated the effects and advantages of using the full color information within our variational framework. The results show that in many cases brightness is not enough to capture important boundaries, especially from object recognition perspective. The use of color information significantly enhances the detection of salient object boundaries.

Chapter 5

Graph Partitioning Active Contours (GPAC)

The (geometric) active contour methods (ACM) [34, 11, 41, 12, 14] and graph partitioning methods (GPM) [60, 57, 58, 2, 111, 59] have become increasingly popular in the last decade. A key advantage is that both these methods result in closed object boundaries. Closed boundaries are desirable both for human perception and machine recognition [112-114].

ACM can be used to integrate both curve (line) processes [115] and regions (points) [11] concurrently while staying a closed contour in the process. The evolution of the curves can be controlled by segmentation cost functions, external force fields and geometric forces such as curvature flow and constant expansion forces (e.g., balloons, see [39]). Based on the driving force behind curve evolution, ACM is divided into two groups, edge-based and region-based ACM. Edge-based ACM attempt to fit an initial curve to its surrounding edges as best as possible.

Usually the results are highly dependent on where the curve is initialized. On the other hand, region-based ACM use regional features of the interior and the exterior of the curve and has been shown to be not as dependent on the initial contour as their edge-based counterparts [116, Chapter 4],[117]. Local minima can be easily avoided by initializing a multi-part curve covering the whole image domain in a grid fashion. Previous work on region-based ACM segmented the images by modeling them as piecewise constant [11], piecewise smooth functions [12], by maximizing separation of the mean or variance of neighboring regions [14], or by clustering the histogram first to estimate region statistics offline and then tuning the ACM to these statistics [41]. Most of these methods are based on statistics of unknown regions and make a priori assumptions about the image characteristics.

Most segmentation techniques can be characterized as edge-based or region-based. Elegant methods for integrating edge and region information exist [10] but these are difficult to solve analytically and are computationally expensive. In this chapter we present a region-based ACM framework that utilizes cost functions using pairwise similarities. This kind of pairwise similarity metrics are most commonly utilized by region-based GPM in their segmentation cost functions. In this chapter, we will show ways of minimizing pairwise similarity based cost functions within the curve evolution framework. We will identify the related problems and propose corresponding solutions. We will also show connections and comparisons to various GPM. For example, the analogous energy functional of the minimum cut criteria of GPM can be written in continuous domain as:

$E = \int_{R_i(C)} \int_{R_o(C)} w(p_1, p_2) dp_1 dp_2$, where R_i and R_o are the interior and the exterior of a closed curve, and $w(p_1, p_2)$ is a similarity measure. We will also discuss several different ways of integrating the edge information.

Variational methods has defined various cost functions for the task of image segmentation in the past. A popular tool for the (local) minimization of some of these cost functions is the curve evolution framework. These variational cost functions can be roughly categorized as contour modeling, region modeling or a combination of these. An example for a contour modeling cost function is the one proposed by Caselles et al. [34] for the geodesic active contour framework. The cost is defined along a curve and minimized by evolving the curve in the normal direction.

$$\underset{C}{Min}\, E = \oint g(C) ds \tag{5.1}$$

where $g = 1/(1 + |\nabla \hat{I}|)$ and $\hat{I}$ is the Gaussian smoothed image. A well known example for the region modeling cost function is the Mumford-Shah functional [10]. A simplified version of this functional, which models the image with piecewise constant functions, has been minimized within the curve evolution framework by Chan et al. [11]:

$$\underset{C}{Min}\, E = \iint_{R_i} (I - c_1)^2 + \iint_{R_o} (I - c_2)^2 \tag{5.2}$$

where R_i corresponds to the interior and R_o corresponds to the exterior of the curve C.

In this chapter, we introduce a new class of variational cost functions that are based on pairwise similarities or dissimilarities between points. The most basic version of such cost function is:

$$\underset{C}{Min}\, E = \iint_{R_i} \iint_{R_o} w(p_1, p_2) dp_1 dp_2 \tag{5.3}$$

where $w(p_1, p_2)$ is a metric for the similarity between the points p_1 and p_2. We will use the notation $w(p_1, p_2)$ for representing both similarity and dissimilarity measures within this chapter and the meaning should be clear from the context. If w is a dissimilarity measure then (5.3) is maximized. The objective of minimizing (5.3) is to minimize the similarity between the regions R_i and R_o. We will show later in Section 5.3 that (5.3) also maximizes the similarity within the regions R_i and R_o.

Main contributions include:

- Introduction of new class of variational cost functions that are based on pairwise similarities. Since such cost functions are popular for GPM, we are able to combine and integrate many novel ideas from both variational and graph partitioning methods to create better techniques (Sections 5.2, 5.3, and 5.4).

- Steepest descent minimization of various pairwise similarity based cost functions within the curve evolution framework. Due to the complexity of the cost functions we introduce in this chapter, minimization of such cost functions has not been attempted within a variational framework (Sections 5.2).

- Minimization frameworks for pairwise similarity based cost functions turn out to be computationally very expensive. Due to the excessive memory and CPU requirements, naive implementations are not practical even on high end workstations. We introduce novel numerical methods for efficient implementation of the curve evolution techniques that are derived for the minimization of pairwise similarity based cost functions (Section 5.7).

- Novel strategies for integrating edge information (Section 5.5), multi-region segmentation (Section 5.6), extensions to multi-scale segmentation (Section 5.6), interactive segmentation (Section 5.9), and a framework to balance precision vs. computational complexity (Section 5.6).

5.1 Comparison of ACM and GPM

The cost functions we introduce in this chapter have been commonly used within the graph partitioning framework. One of the common techniques for minimizing the cost functions in GPM is by finding the appropriate clusters (regions) of graph nodes (e.g. pixels). Others include finding cycles (closed contours) on which a certain cost function is minimized [59]. A popular technique for clustering graph nodes is using the graph spectrum as proposed by normalized cuts [57] and average cut [58] methods.

There are several advantages of ACM compared to spectral GPM.

- The energy functional is minimized in the continuous domain. This makes efficient use of the rich theory developed in the continuous domain possible.

- Edge information can be directly integrated either into the energy functional or into the curve evolution.

- The evolution of curves creates an iterative process. During the evolution, various conditions can be checked or integrated such that the curve's evolution is adjusted accordingly. This introduces added flexibility to our framework.
- Multiple curves can be coupled and can interact with each other during evolution.

On the other hand, the advantage of spectral GPM has been the use of powerful cost functions that are based on a pairwise similarity metric that has been shown to be successful in segmenting complex natural images. The performance and characteristics of these cost functions are rigorously analyzed both analytically using statistical models and empirically on different classes of images [61] (see [61] for more references on this topic). For region-based ACM, we are not aware of similar results. Most of the techniques are only demonstrated by their performance on a limited number of images. Another positive for GPM is that GPM search for an approximate global minimizer as opposed to the steepest descent method used in ACM. Several GPM methods [118, 59], which are edge-based, compared their methods to edge-based ACM and pointed out the advantages of their global minimization properties.

In this chapter we propose a region-based ACM. While edge-based ACM are highly dependent on the initial location of the curve and find a local minimum close to the initial location of the curve, region-based ACM behave quiet differently from their edge-based counterparts. Even though a local minimization method is used, region-based ACM utilize region features calculated on the whole image as opposed to edge-based ACM, where an edge function is generated by local filtering. Various instantiations of the curve give similar results at convergence. This can be thought of as local minimization of a cost function that is defined using global features. On the other hand, GPM such as normalized cuts [57] utilize (approximations of the) global minimization techniques on a cost function that is defined using local features (pairwise similarities are restricted to local neighborhoods).

ACM methods use theoretically well established numerical methods, such as level set methods [32, 33], for their implementation. Issues such as discretization of continuous solutions or edges on image boundary are automatically taken care of. Solutions of cost functions in the continuous domain are also proposed by GPM methods [59] but the issues related to discretization needs to be taken care of separately as the GPM numerical algorithms are usually developed for the discrete domain.

5.2 Curve Evolution based on Minimum Cut Criterion

We will now show a curve evolution solution to the minimum cut problem, which originates from the problem of graph partitioning (Section 2.4). Minimum cut criteria can be written as an energy functional in continuous domain. In this case, the problem is formulated as the partitioning of an image with a curve, as opposed to a graph cut. One possible difference compared to graph partitioning approach is that the partitions of ACM are not necessarily connected components (a curve can split into multiple curves), which is sometimes enforced as a constraint in GPM.

Consider the following cost function that is utilized by minimum cut technique, which is a graph partitioning method.

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v) \tag{5.4}$$

The equivalent energy functional of (5.4) that we would like to minimize is then written as:

$$E = \iint_{R_i(C(t))} \iint_{R_o(C(t))} w(p_1,p_2) dp_1 dp_2 \tag{5.5}$$

where C is a curve, t is the time parameter of the evolution of C and R_i and R_o are the interior and the exterior of this curve. We solve this minimization problem using steepest descent method where we instantiate a curve and evolve this curve towards the minimum.

Theorem 5.1 *Let $\vec{N}$ be the outward normal to the curve C. The curve evolution equation that corresponds to the steepest descent minimization of (5.5) is:*

$$\frac{\partial C}{\partial t} = \left(\iint_{R_i(C)} w(c,p)dp - \iint_{R_o(C)} w(c,p)dp \right) \vec{N} \tag{5.6}$$

Proof:

To find the steepest descent equations, we need to calculate the first variation of (5.5). Before attempting this, first we calculate the first variation of a functional $M = \iint_{R_i(C)} G(X,t)dX$, where X is a point in 2-D. We will then use this to minimize (5.5). Let us write M as:

$$M(t'(t),\tau(t)) = \iint_{R_i(\vec{C}(t'))} G(X,\tau)dX \tag{5.7}$$

where $\tau(t) = t$, $t'(t) = t$, $C = C(t,p)$ is a closed curve and $p \in [0,1]$ is a parametrization of this curve. The first variation of (5.7) with respect to t is:

$$\frac{\partial M}{\partial t} = \frac{\partial M}{\partial t'}\frac{\partial t'}{\partial t} + \frac{\partial M}{\partial \tau}\frac{\partial \tau}{\partial t} \tag{5.8}$$

We first calculate the second term, then we will calculate the first term.

$$\begin{aligned} \frac{\partial M}{\partial \tau} &= \frac{\partial}{\partial \tau}\iint_{R_i(\vec{C}(t))} G(X,\tau)dX \\ &= \iint_{R_i(\vec{C}(t))} \frac{\partial}{\partial \tau}G(X,\tau)dX \end{aligned} \tag{5.9}$$

Before calculating the first variation of M with respect to t', we write M as a boundary integral using the divergence theorem. To do this, we define a vector $\vec{S}$ as:

$$\vec{S} = \begin{bmatrix} \frac{1}{2}\int\limits_0^x G(\lambda,y)d\lambda \\ \frac{1}{2}\int\limits_0^y G(x,\lambda)d\lambda \end{bmatrix} \tag{5.10}$$

As it can be seen, divergence of $\vec{S}$ is equal to G: $\nabla \cdot \vec{S} = G$. Using the divergence theorem, we can write M as:

$$\begin{aligned} M &= \iint_{R_i(\vec{C}(t'))} \nabla \cdot \vec{S}\; dX \\ &= \oint_C \left\langle \vec{S}, \vec{N} \right\rangle ds \end{aligned} \tag{5.11}$$

where $\vec{N}$ is the *outwards* normal vector of the curve and $\langle , \rangle$ denotes the scalar product. Derivation of the first variation of (5.11) with respect to t' has been given by Zhu et al. [40, Appendix], Tsai [116, Appendix A] and Vasilevskiy et al. [119] independently. For completeness purposes, we also include our version of the solution to this problem in Appendix B. The first variation of (5.11) with respect to t' is then:

$$\frac{\partial M}{\partial t'} = \oint_{\vec{C}(t')} \left\langle \vec{C}_{t'}, G\vec{N} \right\rangle ds \tag{5.12}$$

where C_t corresponds to the derivative of C with respect to t. Combining (5.8), (5.12), and (5.9), we find that

$$\frac{\partial M}{\partial t} = \oint_{C(t)} \left\langle C_t, G\vec{N} \right\rangle ds + \iint_{R_i(C)} \frac{\partial}{\partial t} G(X,t) dX \tag{5.13}$$

Now, going back to the problem of calculating the first variation of (5.5), we utilize the result from (5.13) to solve this problem. If we select $G(X,t) = \iint_{R_o(C)} w(X,Y)dY$ within (5.7), then (5.5) becomes equivalent to (5.7). Using (5.13), we can write the first variation of (5.5) as:

$$\begin{aligned}
\frac{\partial E}{\partial t} &= \oint_C \left\langle C_t, \left[\iint_{R_o(C)} w(c,p_2)dp_2\right] \vec{N} \right\rangle ds \\
&\quad + \iint_{R_i(C)} \frac{\partial}{\partial t} \left[\iint_{R_o(C)} w(p_1,p_2)dp_2\right] dp_1 \\
&= \oint_C \left\langle C_t, \left[\iint_{R_o(C)} w(c,p_2)dp_2\right] \vec{N} \right\rangle ds \\
&\quad + \iint_{R_i(C)} - \oint_C \left\langle C_t, w(p_1,c)\vec{N} \right\rangle ds \; dp_1 \\
&= \oint_C \left\langle C_t, \left[\iint_{R_o(C)} w(c,p)dp - \iint_{R_i(C)} w(c,p)dp\right] \vec{N} \right\rangle ds
\end{aligned} \tag{5.14}$$

where c is a point on the curve C. In these calculations, we used the fact that $w(p_1,p_2)$ is a symmetric function and is not a function of t. Thus the first variation of G can be calculated as $\partial G/\partial t = -\oint_{C(t)} \left\langle C_t, w\vec{N} \right\rangle$. When integrating on R_o, the normal vector is in the opposite direction, hence the minus sign. From (5.14) we can see that E decreases fastest when:

$$\frac{\partial C}{\partial t} = \left(\iint_{R_i(C)} w(c,p)dp - \iint_{R_o(C)} w(c,p)dp \right) \vec{N} \qquad (5.15)$$

This concludes the proof. ■

This result mainly means that each point c on the curve C is compared to the interior and the exterior of the curve. The result in (5.15) can be visualized as the competition of the background and the foreground. Note that the theory for this result and related GPM and ACM research assume that the image consists of a foreground and a background. We will discuss extensions to multi-region segmentation later in Section 5.6.

5.3 Maximum Cut and Region Stability

As discussed in Section 2.4, one problem with the minimum cut criterion is that it favors cutting small partitions. This is so, because for small partitions, the total sum across the cut is small. Our minimum cut framework that is introduced in 5.2 also inherits this problem. We will address this problem in this section. Before addressing the problem of favoring small partitions, we discuss another flaw of the original formulation.

Another problem with the minimum cut framework is illustrated in Fig. 5.1(a). Fig. 5.1(a) shows a black object against a white background. Let the similarity measure be $w(p_1, p_2) = \exp(\frac{|I(p_1)-I(p_2)|}{\sigma_I})$ where I corresponds to pixel intensities. Fig. 5.1 shows results for two different instantiation of the curves. In Fig. 5.1(a), a more balanced curve is initialized, where the interior of the curve mostly consists of the foreground object and the exterior consists of the background. The curve evolution finds the correct result easily.

In Fig. 5.1(c), instantiation of a smaller curve within the foreground shows one of the problems in using (5.15). The similarity of each point on the curve to both the interior and exterior of the curve is summed and compared according to (5.15). In this example, there are more black points outside the curve than there are inside. So, the total similarity to R_o is higher, making $\frac{\partial C}{\partial t}$ a negative number. This causes the curve to shrink and disappear (Fig. 5.1(d)), which is not the ideal result. The reason for this behavior is mainly the choice of the similarity measure. In this example we chose the similarity measure as originally proposed in [60, 57]. On the other hand, choosing the similarity measure as $w(p_1, p_2) = -|I(p_1) - I(p_2)|$ would help fix this problem[1]. Unfortunately, defining the highest possible similarity between pixels as 0 is counter-intuitive.

Let us define a dissimilarity measure instead of a similarity measure. Consider

$$w(p_1, p_2) = |I(p_1) - I(p_2)| \tag{5.16}$$

as a dissimilarity measure, where similar pixels have 0 dissimilarity whereas a jump in intensity values cause a high dissimilarity. Based on this definition, we also change the segmentation criterion from minimizing the graph cut to maximizing the cut in (5.5), which corresponds to maximizing the dissimilarity across the cut. We call this framework as *maximum cut*. Maximum cut fixes both of the

[1]This type of similarity function is proposed in [2, Section 5.2] for ratio cut in a different context.

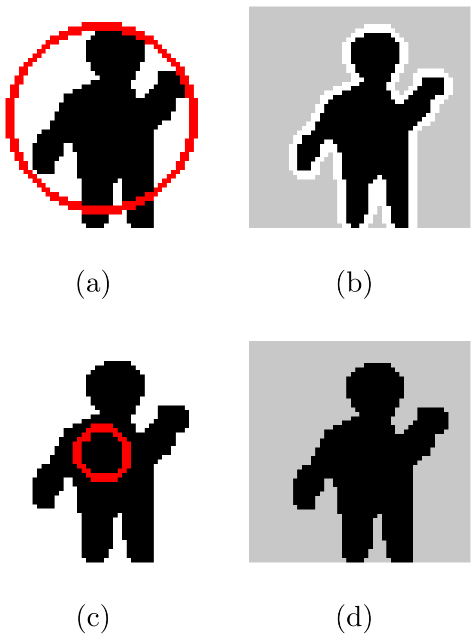


Figure 5.1: a) A large balanced curve initialized. b) Foreground object captured correctly. c) Small curve initialized within the foreground. d) Initial curve shrinks and disappears

shortcomings observed with the minimum cut. The integral in (5.5) is maximized when there are as many connections as possible, which encourages larger partitions as opposed to the minimum cut's behavior of favoring small isolated regions. We also observe that any size or location for the curve instantiation converges to the ideal result in Fig. 5.1. Since the cost function is maximized as opposed to being minimized, the new evolution equation is the negative of (5.15).

$$\frac{\partial C}{\partial t} = \left(\iint_{R_o(C)} w(c,p)dp - \iint_{R_i(C)} w(c,p)dp \right) \vec{N} \tag{5.17}$$

Similar to maximum cut framework, one of the objectives of normalized cut [57] (see Section 2.4) is to fix the behavior of favoring small regions in minimum cut. Even though cost function for maximum cut is different from the cost function used in normalized cuts, there is some parallelism between them. To clarify this consider the following intra-region dissimilarity:

$$E_2 = \iint\limits_{R_i} \iint\limits_{R_i} w(p_1,p_2)dp_1dp_2 + \iint\limits_{R_o} \iint\limits_{R_o} w(p_1,p_2)dp_1dp_2 \tag{5.18}$$

where $w(p_1, p_2)$ is a dissimilarity measure. The objective is to minimize the intra-region dissimilarity. If we solve the curve evolution equations for minimizing E_2, we observe that they are exactly the same as (5.17). This is not surprising since $E_2 = \iint_R \iint_R w(p_1,p_2)dp_1dp_2 - 2E = C - 2E$, where $R = R_i \cup R_o$, and C is a constant. We can conclude that using a dissimilarity measure encourages similarity within the partitions while discouraging inter-region similarity. A parallel argument is made for normalized cuts based on the region associations, see (2.24).

In Fig. 5.2, we corrupt the binary image with Gaussian noise and apply the maximum cut segmentation. As can be seen, for two different curve instantiations, several one pixel wide noisy regions are captured. Note that the white points in 5.2(b) and (d) correspond to the curve and all boundaries together correspond to a single curve over which the cost function is optimized. GPM usually get around this problem by enforcing region connectivity, decreasing similarity with spatial distance and size constraints in its cost functions [2]. ACM solves this problem by adding a curvature flow component, which is a geometric component that smooths the curve at each iteration. Curvature flow can also formulated as a length minimizing flow, which means that it disfavors splitting of the curve to small one pixel wide boundaries. The new evolution equation can be written as:

$$\frac{\partial C}{\partial t} = \left(\iint_{R_o(C)} w(c,p)dp - \iint_{R_i(C)} w(c,p)dp \right) \vec{N} - \gamma \kappa \vec{N} \tag{5.19}$$

where γ is a constant and κ is the curvature of the evolving curve.

Fig. 5.3 demonstrates curve evolution for four different types of initializations of the curves. All four of these evolutions are conducted using (5.19). This example illustrates that (a) maximum cut is independent of the curve instantiation, and (b) that curvature-based flow increases robustness of the curve evolution under noisy conditions.

Normalized Maximum Cut

One of the advantages of using geometric ACM is that we can introduce geometric properties or constraints into the curve evolution equation without changing or re-solving the energy minimization problem. Since curve evolution is an iterative process, we can add normalization for the integrals in (5.19) by dividing them with their corresponding areas. We call this setup as *normalized maximum*

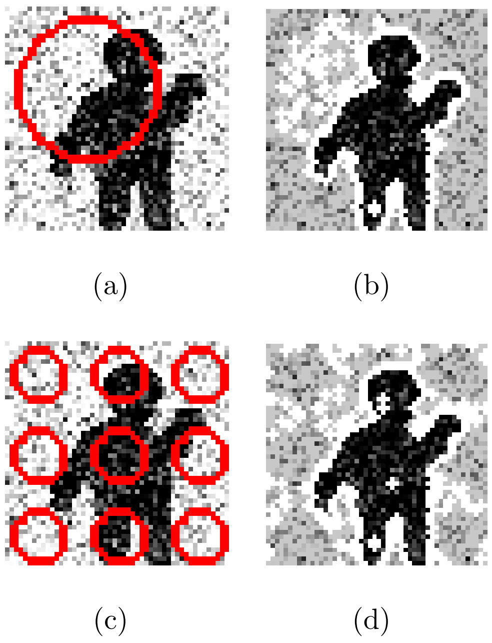


Figure 5.2: Demonstration of how noise can effect the curve evolution. a) Single curve initialized overlapping both the foreground and the background. b) Corresponding curve evolution result (all white points correspond to the curve). c) A multipart curve initialized in a grid fashion. d) Corresponding curve evolution result (all white points correspond to the curve). The curve evolution on a noisy image splits the curve into many small pieces.

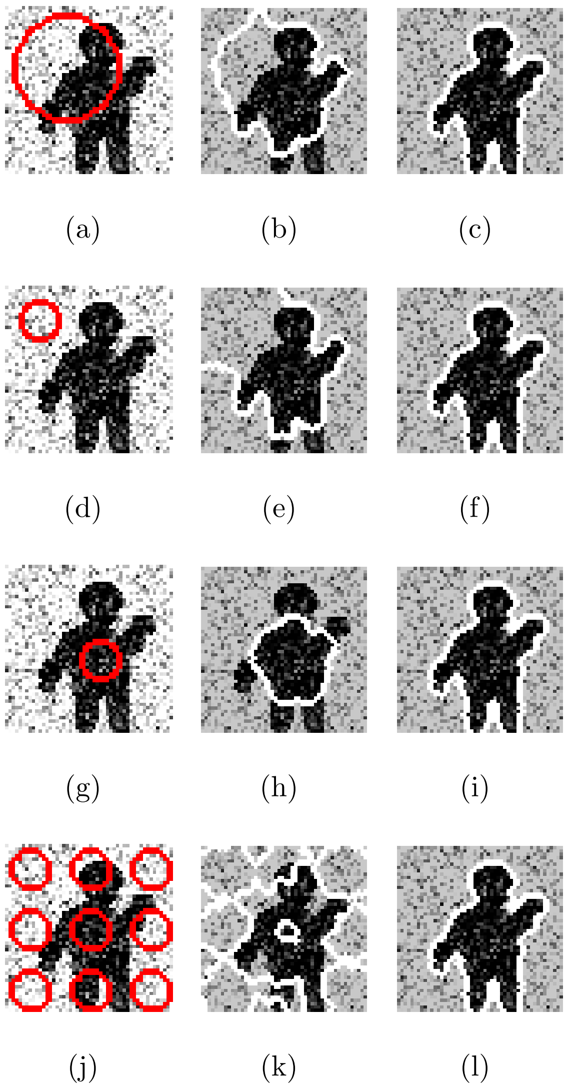


Figure 5.3: Three figures on each row, where each row corresponds to maximum cut evolution under 4 different type of curve instantiations. First column corresponds to various initializations of the curve. Second column shows a state of the curve during the evolution. Third column shows the segmentation result at convergence. This example demonstrates the stability of maximum cut framework with respect to curve instantiation and noise.

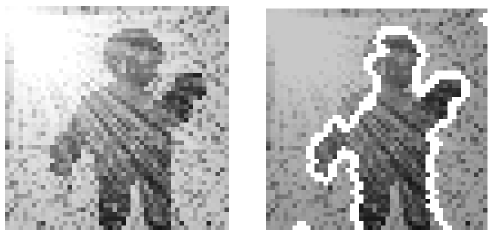

(a) (b)

Figure 5.4: Segmentation of an image corrupted by Gaussian noise and applied an illumination effect. a) Original image. b) Corresponding curve evolution result using normalized maximum cut segmentation.

cut. The evolution equations become:

$$\frac{\partial C}{\partial t} = \left(\frac{1}{A_o} \iint_{R_o(C)} w(c,p)dp - \frac{1}{A_i} \iint_{R_i(C)} w(c,p)dp \right) \vec{N} - \gamma\kappa\vec{N} \tag{5.20}$$

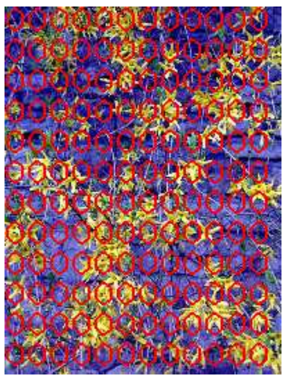
This shows the flexibility of active contour framework compared to GPM. Fig. 5.4 shows normalized maximum cut segmentation on an image that is corrupted by Gaussian noise and applied an illumination effect to the top left corner.

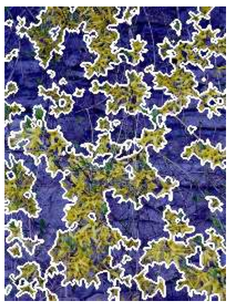
Fig. 5.5 shows both maximum cut and normalized maximum cut segmentation results on a flower image using color information. This is not an easy image for edge-based methods since there are many edges within the foreground and significant clutter in the background. A Gaussian smoothing operator would also cause blurring of the edges of the foreground due to the spread out structure of the foreground. Results for both maximum cut and normalized maximum cut are able to segment the foreground from the background successfully. In general we observed that normalized maximum cut gives slightly better results. Main reason for this is that maximum cut favors equal size regions. For images where the foreground is larger or smaller than the background, it is more efficient to use normalized maximum cut. In our experiments, one multi-part curve with 144 sub-parts is initialized uniformly over the image. We will use this type of initialization for the rest of the chapter. Note that if two sub-parts of the curve touch each other, their boundaries will merge. Level set methods (See Section 2.2.4), which is the numerical method used in all the implementations in this chapter, is able to handle merging and splitting of the curves naturally and also automatically keep track of what is the interior and what is the exterior of the curve.

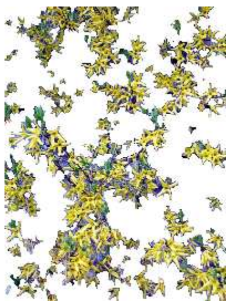
5.4 Curve Evolution for Other Pairwise Similarity Based Cost Functions

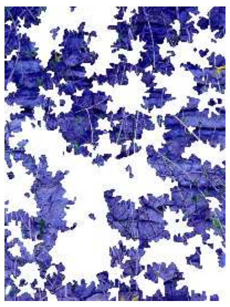
One of the advantages of our variational framework is that the theory developed so far (including the implementation issues that are discussed in 5.7) can be easily adapted and applied to other cost functions that are based on pairwise (dis)similarities. To demonstrate this aspect of our framework, we now derive curve evolution equations for various GPM-based cost functions that address several different problems. References to GPM such as average cut and normalized cut are used to indicate the cost functions associated with these methods, but not to refer to the GPM-based solutions of these problems. This section shows that the proposed curve evolution framework that we introduced in this chapter is general and not specific to a certain cost function.

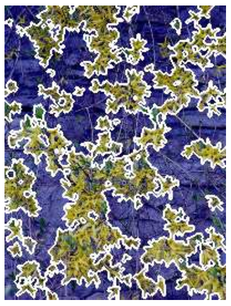
(a)

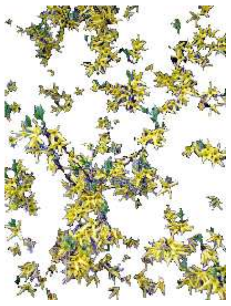

(b)


(c)


(d)


(e)


(f)


(g)

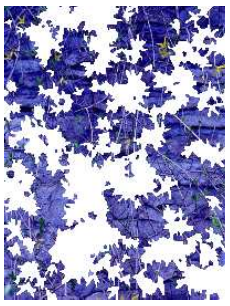

(h)

Figure 5.5: a) Original image (266x200), b) initial curve consisting of 144 sub-parts, c) maximum cut segmentation and corresponding, d) foreground, e) background, f) normalized maximum cut segmentation and corresponding, g) foreground, h) background.

5.4.1 Curve Evolution based on Boundary-normalized Cut

Boundary-normalized cut that we will introduce here is a way of normalizing the cost function by the boundary length between R_i and R_o. Recall that the maximum cut criteria favors having a large number of connections across the cut, which corresponds to a longer boundary length. So, it might be of interest to investigate a boundary-normalized cost function. Boundary-length normalization is also used by several GPM [59, 2]. We can write the corresponding cost function as:

$$E = \frac{\iint_{R_i(C)} \iint_{R_o(C)} w(p_1, p_2) dp_1 dp_2}{\int_0^1 |C_q(q)| dq} \tag{5.21}$$

Let $K = \iint_{R_i(C)} \iint_{R_o(C)} w(p_1, p_2) dp_1 dp_2$ and $L = \int_0^1 |C_q(q)| dq$. The first variation of $E = K/L$ can be calculated as $E' = (K'L - KL')/L^2$. We already calculated K' in Section 5.2. The solution of L' is previously studied [120] and it can be shown that $\partial L/\partial t = \oint_C \left\langle C_t, \kappa \vec{N} \right\rangle ds$. Based on these calculations, the evolution equation that maximizes (5.21) can be written as:

$$\frac{\partial C}{\partial t} = \frac{1}{L} \left(\iint_{R_o(C)} w(c, p) dp - \iint_{R_i(C)} w(c, p) dp \right) \vec{N} - \frac{K}{L^2} \kappa \vec{N} \tag{5.22}$$

The boundary length L is available during each reinitialization of the narrow band used in Level Set methods (See Sections 5.10 and 2.2.4 for more information). Calculating K can be expensive on the full image grid, but this value can be approximated on a low resolution grid for computational efficiency.

This result seems to be conceptually similar to the idea of introducing a curvature-based component as done in (5.19). On the other hand, the effects of the terms such as L and $\frac{K}{L^2}$ are not clear. In the previous section, boundary-based constraint is introduced as an additional term in the cost function, whereas in (5.21) the cost function is normalized by the curve length. We can also write a version of (5.22) that is normalized by area:

$$\frac{\partial C}{\partial t} = \frac{1}{L}\left(\frac{1}{A_o}\iint_{R_o(C)} w(c,p)dp - \frac{1}{A_i}\iint_{R_i(C)} w(c,p)dp\right)\vec{N} - \frac{K}{L^2}\kappa\vec{N} \qquad (5.23)$$

5.4.2 Curve Evolution based on Average Cut Criterion

In this section we find the descent equation for the average cut cost function:

$$Acut(A,B) = \frac{cut(A,B)}{|A|} + \frac{cut(A,B)}{|B|} \qquad (5.24)$$

This cost function can be written in continuous domain as:

$$E = \frac{K}{\int_{R_i} dX} + \frac{K}{\int_{R_o} dX} = \frac{K}{A_i} + \frac{K}{A_o} \qquad (5.25)$$

where K is defined in previous section and X is a point in 2-D. Note that the integrations in the denominator can also be done over a function $f(X)$, which might contain modeling information about the objects in the image.

Following a similar calculation as in Section 5.4.1, we can see that $E' = (K'A_i - KA_i')/A_i^2 + (K'A_o - KA_o')/A_o^2$. We calculated K' in Section 5.2. A_i' and A_o' can also be calculated as special cases of (5.13) where $G(X) = 1$. This gives $\partial A_i/\partial t = \oint_C \left\langle C_t, \vec{N}\right\rangle ds$. For A_o the normal vector is in the opposite direction, which introduces a minus sign. The combined flow equation can then be written as:

$$\frac{\partial C}{\partial t} = \left[\left(\frac{1}{A_o} + \frac{1}{A_i}\right)H + \left(\frac{1}{A_o^2} - \frac{1}{A_i^2}\right)K\right]\vec{N} \qquad (5.26)$$

where $H(c) = \iint_{R_o(C)} w(c,p)dp - \iint_{R_i(C)} w(c,p)dp$

5.4.3 Curve Evolution based on Normalized Cut Criterion

One of the popular ways to normalize the cost of a graph cut is the normalized cut framework. The graph theory version of this cost function is given in (2.22). The continuous domain equivalent for this cost functions can be written as:

$$E = \frac{K}{B_i} + \frac{K}{B_o} \tag{5.27}$$

where $B_i = \iint_{R_i(C)} \iint_R w(p_1, p_2) dp_1 dp_2$ and R corresponds to the full image domain. B_o is defined similarly. Calculation of the curve evolution equation is straight forward:

$$\frac{\partial C}{\partial t} = \left[\left(\frac{1}{B_o} + \frac{1}{B_i} \right) H(c) + \left(\frac{1}{B_o^2} - \frac{1}{B_i^2} \right) K \cdot Z(c) \right] \vec{N} \tag{5.28}$$

where $Z(c) = \iint_R w(c, p) dp$.

In this section, we have derived curve evolution equations for various pairwise similarity based cost functions, each of which could be useful for different kinds of applications. A possible future direction is an in depth comparison of these curve evolutions that would help us understand the importance of the various terms that have emerged, namely L, L/K^2, K, Z, B_i and B_o.

5.5 Strategies for Integration of Edge Information

So far, we have discussed region-based strategies for segmentation. It is often desirable to integrate the edge information to extract more precise boundaries.

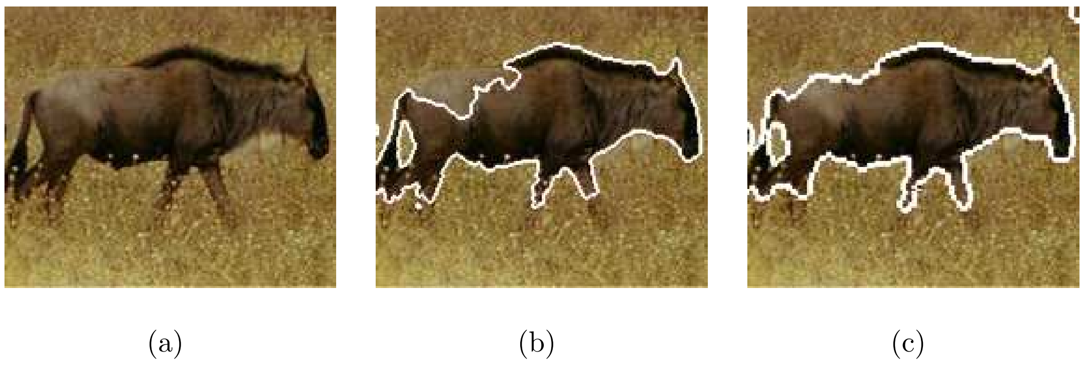


Figure 5.6: a) Original image, b) Segmentation result without edge integration, c)Segmentation result with edge integration.

ACM most commonly utilize an edge function $g = 1/(1 + |\nabla \hat{I}_\sigma|)$, where $\hat{I}_\sigma$ is the Gaussian smoothed image. This edge function is generated from the image through filtering and derivative operators. Another possibility is to create an edge function from a vector field that is derived from the image (Section 3.1). The general characteristic of an edge function is that it is a decreasing function of the edge strength.

One way of integrating edge information is to multiply the curve evolution equations with g. This methodology is used by edge-based ACM. The evolution equation of maximum cut segmentation (5.19) can be written as:

$$\frac{\partial C}{\partial t} = g \left(\iint_{R_o(C)} w(c,p)dp - \iint_{R_i(C)} w(c,p)dp \right) \vec{N} - \gamma g \kappa \vec{N} \tag{5.29}$$

The effect of multiplying with g is that the evolution of the curve will slow down or stop when the curve is aligned with an edge. Fig. 5.6 shows an example of how integrating edge information helps with localization of the boundaries.

Another way to integrate edges is by modifying the cost function for the boundary-normalized cut introduced in (5.21). Instead of just normalizing by the boundary length, we can also normalize by the boundary length weighted by the edge function (a similar normalization is also proposed in [59]).

$$E = \frac{\iint_{R_i(C)} \iint_{R_o(C)} w(p_1, p_2) dp_1 dp_2}{\int_0^1 g(C)|C_q(q)|dq} \tag{5.30}$$

Lets define $L_2 = \int_0^1 g(C)|C_q(q)|dq$. The solution of this cost function is similar to the solution of (5.21). The only difference is that L is replaced by L_2 and the first variation of L is replaced by L_2'. First variation of L_2 has been derived in [34, Appendix B] and is equal to $\partial L_2/\partial t = \oint_C \left\langle C_t, (\nabla g\vec{N} - g\kappa)\vec{N} \right\rangle ds$. Based on these calculations, the corresponding evolution equation can be written as:

$$\begin{aligned} \frac{\partial C}{\partial t} = \frac{1}{L_2} \left(\iint_{R_o(C)} w(c,p)dp - \iint_{R_i(C)} w(c,p)dp \right) \vec{N} \\ - \frac{K}{L_2^2}(g\kappa - \nabla g\vec{N})\vec{N} \end{aligned} \tag{5.31}$$

A third way of using edge information is by utilizing it when defining the similarities between pixels, $w(i,j)$. This approach can be applied to our framework with ease considering that the curve evolution is independent of the (dis)similarity measure. In [121] it has been proposed that, if there are strong edges along a line connecting two pixels (intervening contour), these pixels probably belong to different regions and should be labeled as dissimilar. So, edge information can be integrated by reducing the pairwise similarity of such pixels.

5.6 Multi-region Segmentation

One widely used approach for extending the bi-partitioning presented so far is to continue applying bi-partitioning to each region recursively. Both region-based

ACM [14, 12] and most GPM utilize this simple strategy. All methods proposed in this chapter can use recursive bi-partitioning strategy without any change to the underlying algorithms.

Another multi-region segmentation approach for GPM is to use k-means clustering on the normalized Laplacian matrix of the graph [57]. One problem with this approach is that the number of regions, k, needs to be known beforehand. In [122], k is initially selected very high to generate an over-segmentation of the image. This over-segmentation is then used for coarsening the graph. These new regions are considered as nodes in a new graph and a recursive bi-partitioning is applied to this simpler graph to achieve segmentation.

For ACM, it is more natural to use several independent but coupled curves, all evolving simultaneously and interacting with each other during this evolution (the multi-part curve we previously used in this paper is considered as one curve). Level set methods evolve a 2-D curve by embedding it into a 2-D function $u(x, y)$ as its zero level set. We implement this by defining u on a several pixel wide narrow band around the curve (narrow band level set methods). In this case, u is negative on one side of the curve and positive on the other side. This implicitly defines two regions, interior and exterior of the curve. Triple junctions are not easy to define and implement in this framework. Even though powerful, this makes it difficult to implement multi region cost functions using level set methods.

Recent work of Vese and Chan [13] offers a solution to this problem by using $m = log_2 n$ level set functions to define n regions. Background and different combinations of overlaps of these level set functions are considered as separate regions. Another method for multi-region segmentation using level set methods is geodesic active regions [41]. Geodesic active regions assume that a priori information (probability distributions) about the regions in the image is given. Based on this information, multiple curves, each tuned to different region statistics, are evolved and discouraged from overlapping by slowing them down if they get close to each other. Other approaches include region competition [40], where random seeds are initialized and evolved[2]. No overlaps are allowed. After convergence, regions are merged if this action reduces an energy functional.

In this section, we propose a different way of handling multi-region segmentation. We do not assume any a priori knowledge about the region–this information is automatically discovered by the dissimilarity measure. We do allow overlaps. For transition areas between two dominant regions, it might not be clear which region they belong to. In this situation, it might be better to assign this transition area to both regions by allowing overlaps (After convergence, the boundaries of these regions can be post-processed to find a proper partitioning of the image). Our multi-region approach is based on merging and splitting of regions directly within the curve evolution framework.

The original cost function we defined in (5.5) assumes that there are only two regions in the image. This cost function can be easily extended to handle multiple regions. Let $\{C_i : i = 1 : N\}$ be a set of non overlapping curves and $\{R_i : i = 1 : N\}$ be the regions contained in C_i. Let R_0 be the background. The

[2]Region competition does not use level set methods for its implementation.

multi-region cost function can be written as:

$$E = \sum_{\substack{i,j=0:N \\ i,j \text{ neighbors}}} \iint_{R_i} \iint_{R_j} w(p_1, p_2) dp_1 dp_2 \tag{5.32}$$

Now we consider the initial case where several curves are instantiated on the background. Initially, the only boundaries are between R_0 (background) and the R_i's. The cost function becomes the linear combination of $\iint_{R_i} \iint_{R_0} w(p_1, p_2) dp_1 dp_2$. The corresponding descent equation can be also written as the combination of individual descent equations given in (5.15). At the beginning, all curves are evolving against the background. The interaction between the curves is through the changes to the background by the evolution of each curve. Obviously, this cost function does not hold when two curves touch each other or overlap. Despite this fact, in this situation we continue evolving the curves against the background and ignore the spatial relations between the curves. We let the curves overlap freely. If two curves are significantly overlapping each other, there is a high chance that both of them are targeting regions with similar properties and merging them would be appropriate. If a curve overlaps 30% or more area of another curve, the following condition for region merging is checked for curves C_i and C_j:

$$\left\| \frac{1}{A'_i} \iint_{R'_i} F(p) dp - \frac{1}{A'_j} \iint_{R'_j} F(p) dp \right\| < c_{\text{merge}} \tag{5.33}$$

where $R'_i = R_i - R_i \cap R_j$, $R'_j = R_j - R_i \cap R_j$, A'_i and A'_j are corresponding areas. c_{merge} is a constant threshold and F is an image feature (gray scale, color, or texture depending on the features used in the dissimilarity measure). Curves are merged by removing these two curves and adding a new curve that surrounds $R_i \cup R_j$. After all curves converge, we check each region for homogeneity using the following criterion:

$$\frac{1}{A_i^2} \iint_{R_i} \iint_{R_i} w(p_1, p_2) dp_1 dp_2 > c_{\text{split}} \tag{5.34}$$

where c_{split} is a constant threshold. If this criterion is above a certain threshold for a region, we apply bi-partitioning to this region by initializing a curve within this region.

Algorithm 5.1 Multi-region segmentation algorithm

```
1. Initialize several curves on the image region automatically.
2. Evolve these curves against the background.
3. If a curve overlaps 30% area of another curve, these two curves are merged
   based on the criterion given in (5.33).
4. After all curves converge, the interior of each curve is checked for homogeneity
   using (5.34). Regions that exceed a certain threshold are split into two by
   initializing a curve within them and applying a bi-partitioning.
```

One other advantage of using multiple coupled curves is that this allows us to easily parallelize the computations. Each curve can be evolved in a separate thread where they are synchronized at each iteration or every N iterations.

5.7 Efficient Implementation

As previously mentioned, the integral calculations in (5.17) is usually the bottleneck in terms of computational complexity of the curve evolution. It will be much more costly if we also try to calculate the dissimilarities at each iteration. On the other hand, curve evolution itself can be implemented efficiently and accurately using narrow band level set methods. Since calculating the dissimilarities at each iteration is impractical, we need to calculate the dissimilarity of each pixel to another pixel before the curve evolution starts. Suppose the image is of size $N \times M$, then we need to calculate and keep in memory about $N^2M^2/2$ dissimilarities. This is equivalent of generating a symmetric dissimilarity matrix W with NM rows and columns, where an element at ith row and jth column is $w(p_i, p_j)$. Even for small images, this will become hard to fit into memory and require too many computations. We will address these issues in this section and propose an efficient way of calculating dissimilarities and implementing the curve evolution given in (5.17).

For an efficient implementation of our framework, we create a dissimilarity matrix W' of size $NM \times nm$, where $n \ll N$ and $m \ll M$. W' is an approximation of W and we will demonstrate that segmentation precision is not lost by this approximation. We divide the image into $n \times m$ equal size tiles T_{ij} and average the image features $F(x, y)$ within each tile.

$$F'_{ij} = \frac{1}{A_{ij}} \int_{T_{ij}} F(x, y) dx dy \tag{5.35}$$

The elements of W' are calculated as $\|F(x, y) - F'_{ij}\|$, where the coordinate (x, y) falls into the tile T_{ij}.

Using W', (5.17) can be implemented efficiently. For each point c on the curve (actually for all points in the narrow band), the summation is done over all the tiles whose center falls inside and outside the curve. In doing this, we exclude the tile containing c from the calculations. Assume T_k, $k = 1 : nm$ are the tiles, P_k is the center coordinate of T_k. The difference of integrals in (5.17) can be simplified to $\sum_{k,P_k \in R_o} W'(c,k) - \sum_{k,P_k \in R_i} W'(c,k)$. Using this tile-based framework, integral calculations do not dominate the computational complexity anymore and the cost of curve evolution implementation also becomes important. This gives us an opportunity to speed up our segmentation by replacing our narrow band level set (See Section 2.2.4) implementation with a more efficiently-coded version or by utilizing new types of level set methods as they become available [52].

Figures 5.9 and 5.10 visually show that the precision of the segmentation is not affected by the approximation of W with W'. In all the figures in this chapter, n and m are selected around 15 regardless of the image size. We choose the tiles as squares, where the dimensions of the tiles are $s_x = s_y = \lfloor Width/15 \rfloor + 1$. Then,

$$\begin{aligned} n &= \left\lfloor \frac{Width - 1}{s_x} \right\rfloor + 1 \\ m &= \left\lfloor \frac{Height - 1}{s_y} \right\rfloor + 1 \end{aligned} \tag{5.36}$$

It might seem surprising that even with quantizing one dimension of W, still a precise segmentation can be reached. This can be explained by observing the curve evolution equation (5.17). The energy functional given in (5.5) is symmetric for the dimensions of W. Changing W to W^T will not affect the result. On the other hand, the curve evolution in (5.17) does not have the same symmetry. A single point c is compared to the rest of the points of the image. Coarsening the location of c would have a significant effect on the end result since even a neighbor of c might have a totally different feature value. On the other hand, approximating the summation over all the rest of the points is more robust to errors. This allows us to reduce the resolution of one dimension of W without significantly affecting the end segmentation.

Consider a tile T_j of size N^2 located within the background R_o. Let T_j^s be the pixels inside this tile and c a point on the curve. Based on our simple dissimilarity measure, the normalized dissimilarity of c to T_j is $\frac{1}{N^2}\sum_s \|I(c)-I(T_j^s)\|$ at the full resolution. Using W', an approximation of the same dissimilarity can be written as:

$$\|I(c) - \frac{1}{N^2}\sum_s I(T_j^s)\| = \frac{1}{N^2}\left\|\sum_s I(c) - I(T_j^s)\right\|$$

As it can be seen, if all $I(T_j^s)$ are smaller or larger than $I(c)$, both dissimilarities–at full resolution and after approximation–are equal. For most of the tiles, image features do not vary much within a tile (ignoring outlier points) if the tile is located inside an object. This is not the case for tiles overlapping an object boundary and this might introduce some inaccuracies into the dissimilarity calculation. Also note that, usually false movements of the curve are automatically corrected in the following iterations if c reaches an incorrect region as a result of these errors. Even though we do not pursue in this chapter, size and geometry of the tiles can be selected more adaptively by analyzing the image features. For example, tiles can be analyzed for homogeneity by clustering the histogram of the pixels within the tiles[3] , or an edge detector output can be utilized to find inhomogeneous tiles and maybe split them. Finally, a simple over-segmentation of the image can be used as the tiles so that image features are homogenous within each tile.

[3]Expectation-Maximization (EM) algorithm can be used to find a mixture of Gaussians if the dimensionality of F is not high. If the dimensionality is high, each dimension can be analyzed separately.

5.8 Multi-scale Strategies

For some applications such as content-based image retrieval, the precision of the segmentation is not very important as long as the main regions are roughly identified. Especially if a very large database of images is needed to be segmented, some accuracy might be sacrificed for speed. In the previous section, we proposed reducing resolution of one dimension of W to increase segmentation speed without sacrificing accuracy. It is also possible to reduce the resolution of the other dimension of W. Unfortunately, this will introduce a staircase effect on the boundary. This effect can be compensated by the use of edge information as proposed in equations (5.29) and (5.31) and for the case of small tiles such as 2×2 or 4×4, curvature-based force can easily correct this. Another possibility is to use an over-segmented set of regions as tiles to avoid the staircase effect. So, reducing the size of W' further reduces the complexity of the integral calculations even more. In addition to this, this approximation also has positive effects on the curve evolution itself. We observed that the time step Δt of the iterations can be increased several times without causing instability (this is an heuristic observation and we don't have an analytic proof for this claim). Using averages over tiles might seem to be equivalent of reducing the size of the image. This is not true because curve evolution is conducted on the full image and curvature and the edge information (if utilized) are not the same for different sizes of the image.

The speed of the curve evolution is also proportional to the number of iterations required until convergence is reached. If we were able to instantiate the curve close the the final boundary, segmentation can be achieved very quickly in several iterations. In this regard, the imprecise segmentation we discussed in the previous paragraph can be adapted such that the tile size is slowly reduced as the curve evolves. This will lead the curve first to an approximation of the end segmentation where we switch to the full resolution to extract the precise boundary. This can be thought of as a multi-resolution approach.

Another approach to rapidly evolving the curve towards the final boundary is by using a multi-scale approach. This approach is feasible if edge information is not utilized. First, several low resolution versions of the image are created. We start segmentation at the lowest resolution. After convergence, we scale the resultant curve to the next resolution and continue similarly until segmentation converges at the full resolution. For low resolutions, curvature doesn't need to be used. We do not want to discourage very small regions at the low resolution images. These are not necessarily noise but probably larger and important regions at the full resolution. We also reduce the strictness of the convergence criteria (convergence criteria is discussed in Section 5.10). The result at the low resolution is only the initialization condition for the next resolution, so a full convergence is not required at the intermediate resolutions.

5.9 Interactive Image Segmentation

We now propose an extension to GPAC so that user interaction can be utilized to segment objects of interest in an image. The interaction is based on the user drawing a rectangle mostly covering the object. Cox, et al. [118] proposed an edge-based graph partitioning method for interactive segmentation. In their method the user is asked to draw a bounding box. Then an optimum boundary is searched within this bounding box such that the boundary goes through the strong image edges. This method requires the bounding box to fully contain the object of interest. Our proposed method also requires the user to draw a rectangle around the object of its interest, but it is not required that this rectangle fully contains the object. This application might be useful for very large images such as aerial images, where the objects of interest (lakes, harbors, airports) are relatively small.

Let R_{rect} be the user drawn rectangle. We initialize a multi-part curve as usual, but only within R_{rect}. R_i is defined as the interior of the curves and $R_o = R_{rect} - R_i$. So, we are restricting R_o to within the user drawn rectangle. R_i on the other hand is not restricted to R_{rect} and allows evolving the curve past the rectangle boundary. On the other hand, we don't want the curve to evolve too much outside the boundary because the assumption is that most of the object is contained within the rectangle. So we multiply the curve evolution given in (5.20) by $f(c) = exp(\frac{-dist(c,R_{rect})}{\sigma})$, where $dist(c, R_{rect})$ is the distance of point c from R_{rect} and $f(c) = 0$ if c is located within the rectangle. The curve evolution in this case becomes:

$$\begin{aligned}\frac{\partial C}{\partial t} = f(c)\left(\frac{1}{A_o}\iint_{R_o(C)} w(c,p)dp - \frac{1}{A_i}\iint_{R_i(C)} w(c,p)dp\right)\vec{N} \\ -\gamma\kappa\vec{N}\end{aligned} \tag{5.37}$$

Fig. 5.7 demonstrates that users can easily segment an object of their interest from the image.

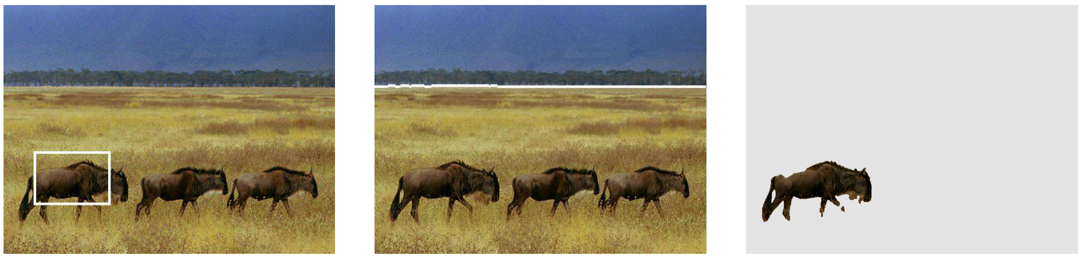


Figure 5.7: a) Original image and the user drawn rectangle, b) Segmentation result without user interaction, c)Segmentation result using the user drawn rectangle.

5.10 Experimental Results

In this section we present experimental results regarding the theory developed in the previous sections. Unless otherwise stated normalized maximum cut framework given in (5.20) and color features are used in the experiments for the segmentation of the images. The reason for choosing color features is that it is easier to visualize and evaluate color segmentation than texture segmentation. We will also demonstrate how utilizing Gabor texture features improves the results compared to using color features alone. For efficiency purposes, the distances between feature vectors are calculated using L_1 distance metric.

Our implementation of Level Set Methods uses the narrow band approach, which is explained in Section 2.2.4 and [33, Chapter 7]. The evolving curve C is embedded into a 2-D function $u(x, y)$, such that $C = \{(x, y)|u(x, y) = 0\}$. This function u is generated and evolved only on a narrow band around the curve instead of the full image domain for efficiency purposes. We select the size of the narrow band as 2 pixels wide and the size of the land mine area (See Section

2.2.4) as 1 pixels wide at both sides of the curve. When the curve reaches the land mine area, narrow band is regenerated from the current curve and the values of u are recalculated on the new narrow band. Even though the specific choice of u mostly does not effect the curve evolution, a popular choice for u is the signed distance function, where each point in the narrow band is assigned a value based on their signed distance from the curve (negative if inside the curve, positive otherwise). Since we are using a very narrow narrow band, we use a different and simpler strategy for the reinitialization of u. We start with the curve and grow the narrow band layer by layer by using the 4-neighbors. First layer consists of the 4-neighbors of the curve points. The second layer is the 4-neighbors of the first layer, etc. Moving from one layer to the next, we increment (decrement) the value of u by 1. The convergence criterion for the curve evolution is also connected to the narrow band approach. Convergence is achieved if the narrow band is not reinitialized for n iterations. This means that the curve moved only within the narrow band during these iterations. In our experiments, n is chosen between 30 and 50.

Fig. 5.8 shows segmentation of a gray scale intensity image of an arterial tree. Segmentation of this image of size 183×163 takes less than 5 seconds on a Pentium 4 2.0 GHertz computer using our unoptimized code written in C#.

Fig. 5.9 and 5.10 show foreground/background segmentation on variety of images[4]. Image resolutions are reduced to 300×200 from their original sizes before segmentation. As can be seen from the figures, regions are precisely segmented

[4]Images on figures 5.10, 5.12, 5.14 and 5.15 and are taken from Berkeley Segmentation Dataset (BSD) training images.

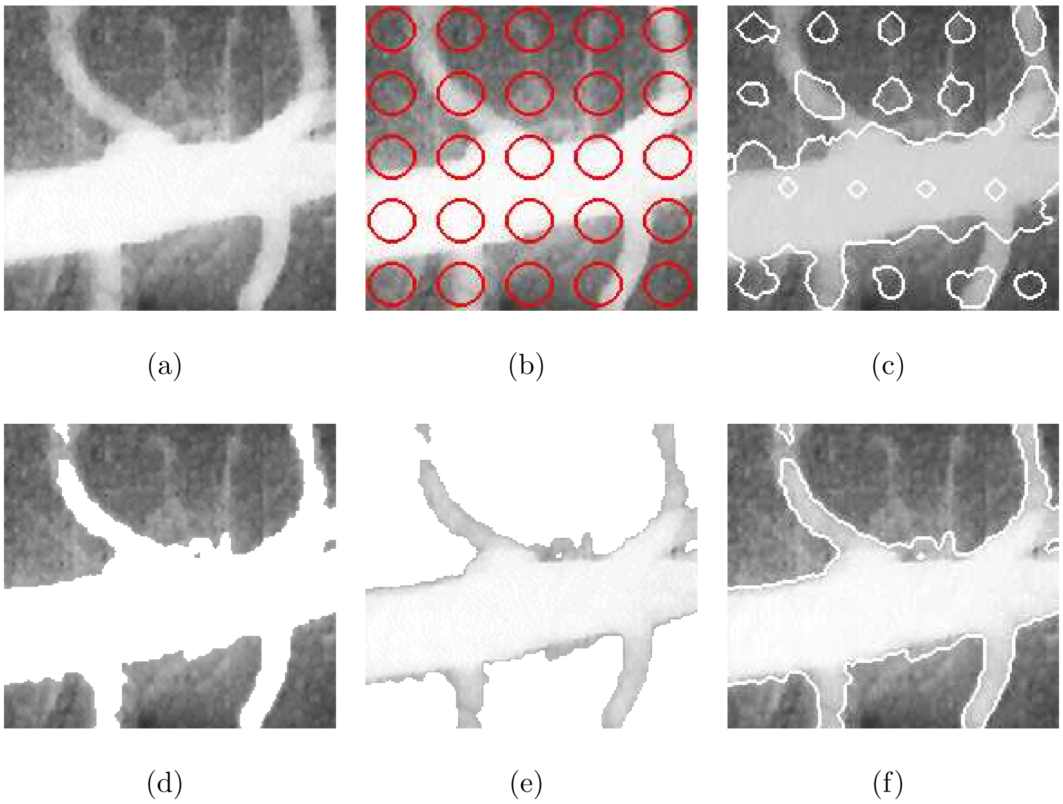


Figure 5.8: Segmentation of an arterial tree image. a) Original image. b) Curve initialization. c) Curve is evolving. d) Segmented background. e) Segmented foreground. f) Segmentation boundary.

despite the fact that we are using the fast scheme from Section 5.7. In these experiments and in the following ones, all parameters are fixed and images are segmented automatically without any manual intervention. Even though this kind of bi-partitioning shouldn't be thought as a full segmentation, it has many applications in various fields ranging from content-based image retrieval to segmentation of biological or medical images.

Color features by themselves might not be able to segment natural images properly. These type of images are usually rich in texture and using texture features will improve segmentation results. Fig. 5.11 shows a comparison of segmentation results using only color features, only texture features and a combination of color and texture features on a bear image. Color features by themselves are not able to extract the boundaries due to the color changes within the head area. On the other hand, the texture of the fur helps finding the precise boundary if texture features are utilized. To calculate the texture features, we filter the image by Gabor filters [123] tuned to certain scales and orientations. In Fig. 5.11, features are calculated at 3 scales and 6 orientations. In Fig. 5.11(d), color and texture features are combined for the segmentation. We do this by creating a feature vector $[\alpha\vec{T}\ (1-\alpha)\vec{C}]^T$, where $\vec{T}$ is the texture feature vector normalized by the average of all the texture feature values and $\vec{C}$ is the color feature vector normalized similarly. α is the weighting between color and texture features, which is selected as 0.7 in Fig. 5.11(d).

In Fig. 5.12, we demonstrate multi-region segmentation using 4 curves on an image of a woman. After automatic merging, interior of the three remaining

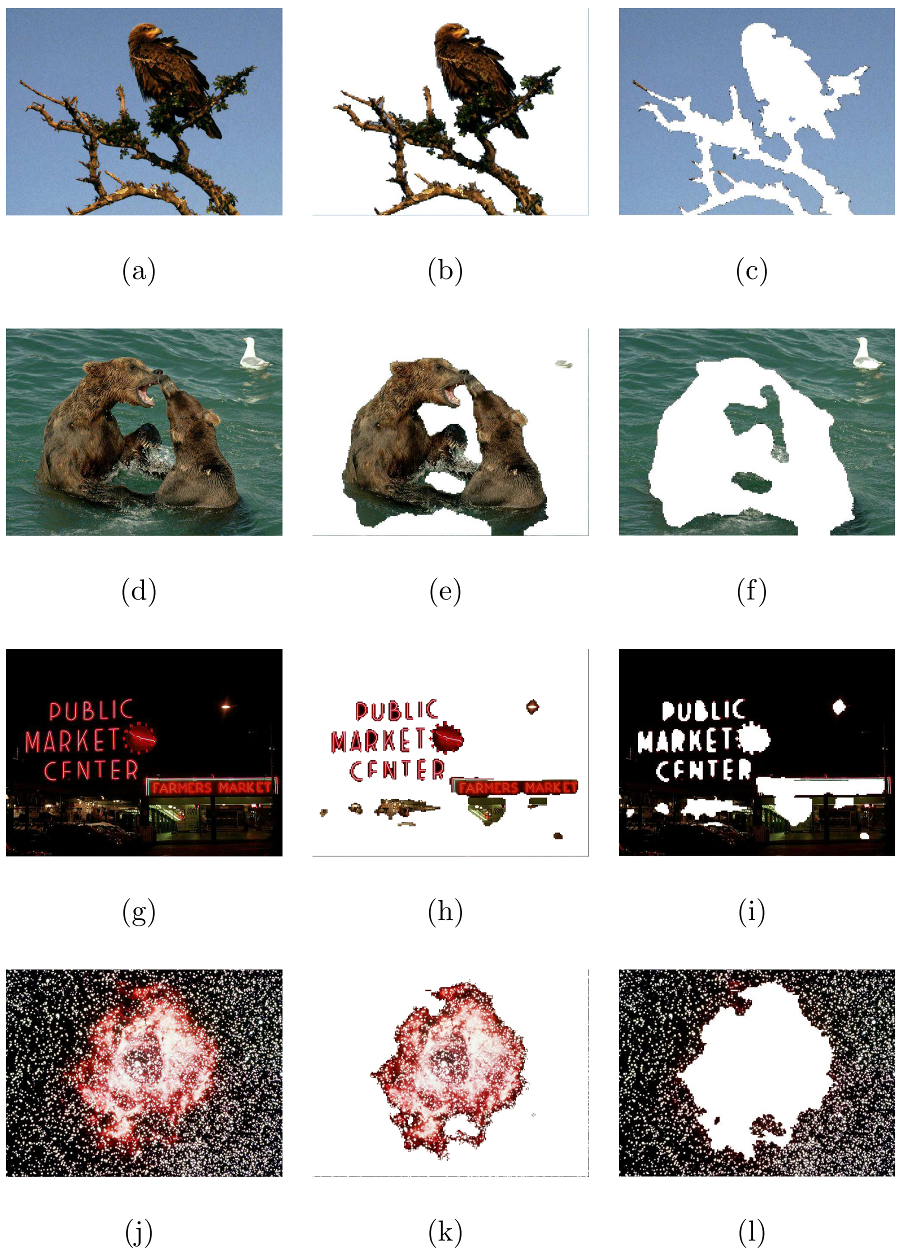


Figure 5.9: Foreground/Background segmentation. First column shows the original images. Second and third columns correspond to foreground and background.

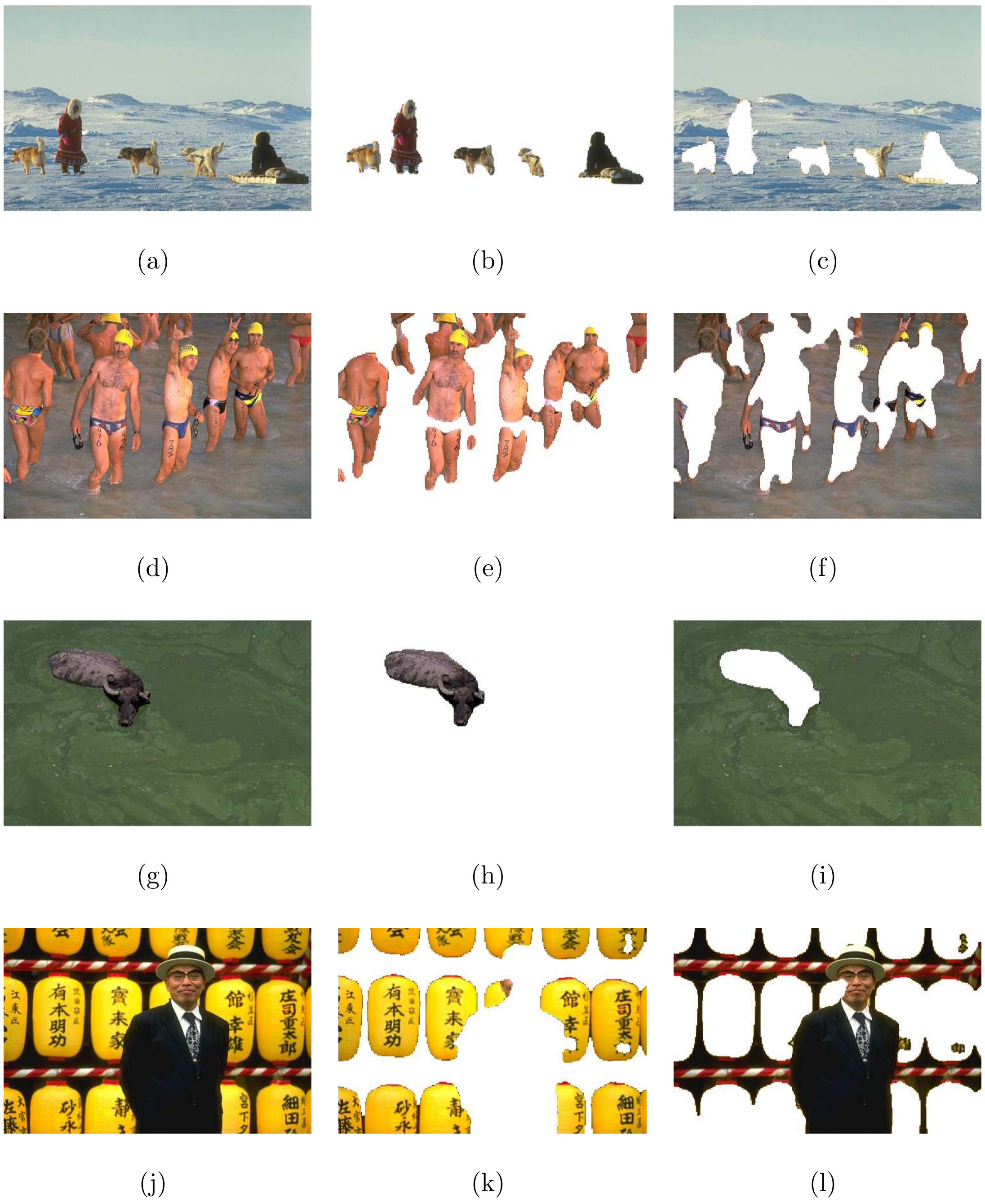


Figure 5.10: Foreground/Background segmentation. First column shows the original images. Second and third columns correspond to foreground and background.

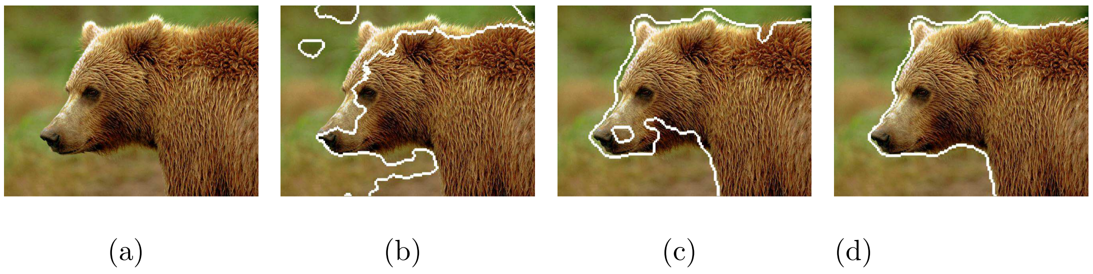
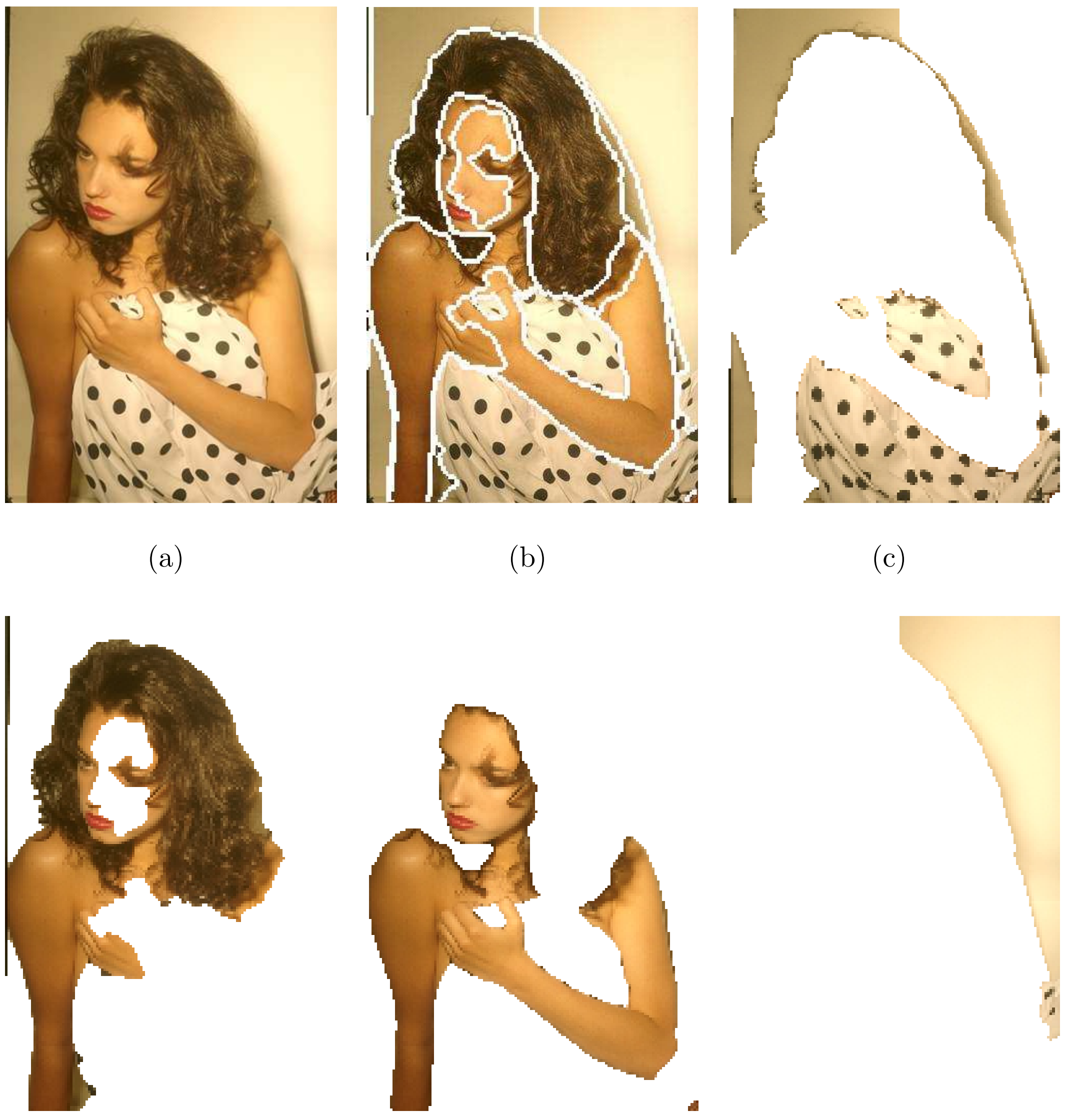


Figure 5.11: Demonstration of improvements using texture feature vectors. a) Original image. b) Color segmentation. c) Texture segmentation using Gabor texture features at 3 scales and 6 orientations. d) Segmentation using texture and color features. Texture features weighted 70% and color features 30%.

curves are given in Fig. 5.12 d-f. As can be seen, the curves are allowed to overlap.

Figures 5.13 and 5.14 show additional results using our novel multi-region segmentation scheme that are based on evolving several coupled curves and allowing them to overlap and merge during evolutions. Each segmentation started with 9 curves and at the end due to merging during curve evolution number of curves is reduced to 2 to 5. Details of this segmentation scheme is given in Section 5.6 and Algorithm 5.1. None of the multi-region segmentation examples we show in this chapter use the splitting step, which is done after the curve evolution finishes. This is because we want these segmentations to reflect the performance of the coupled curve evolution process directly. It might be observed that there are some undersegmented regions, which can be split further through bi-partitioning.

Fig. 5.15 shows a comparison of the foreground/background segmentation against the multi-region segmentation. It can be seen that if the image has more than two main regions, using multi-region segmentation gives a better partitioning of the image, which is to be expected.



(a) (b) (c)

(d) (e) (f)

Figure 5.12: Multi-region segmentation using coupled curve evolution a) Original image. b) Segmentation boundaries. c) Background region. d-f) foreground regions each corresponding to the interior of the curve.

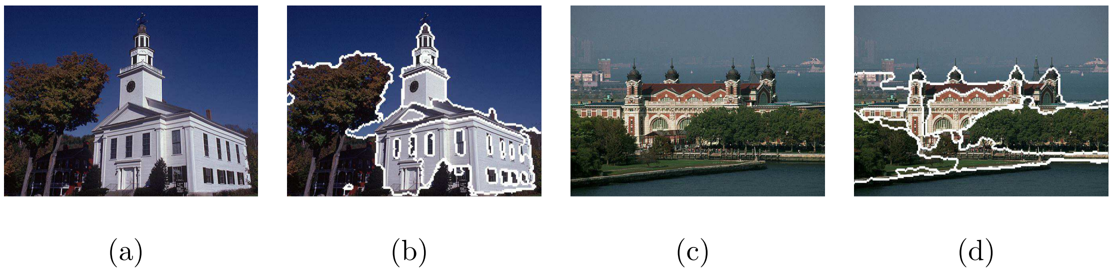


Figure 5.13: Segmentation results for the multi-region approach using coupled curve evolutions. 9 curves are initialized for each image.

5.11 Discussions

Our proposed method, GPAC, is based on pairwise dissimilarities between pixels. The method is quite flexible since the dissimilarity metric can be adapted to the application at hand or to the domain knowledge. This is in contrast with other variational methods where analysis is done at the region level. Usually a region is assumed to be a piecewise constant or smooth area that is corrupted by small gaussian noise. Consider the Fig. 5.16, where the image consists of two regions with same means but different variances. Region-based ACM, such as [14] and [11] assume piecewise constant regions. These methods, without changes to their basic cost functions, would have difficulty segmenting this image. GPAC on the other hand easily segments this image.

In Fig. 5.17 (reproduced from [2]), segmentation result for ratio cut is shown. The result is achieved after 4 bi-partitioning steps. On the same image, our normalized maximum cut framework achieves the same result with one bi-partitioning step. It is debatable if it is better to enforce connected components or not. The answer depends on the application. Curve evolution framework offers more flex-

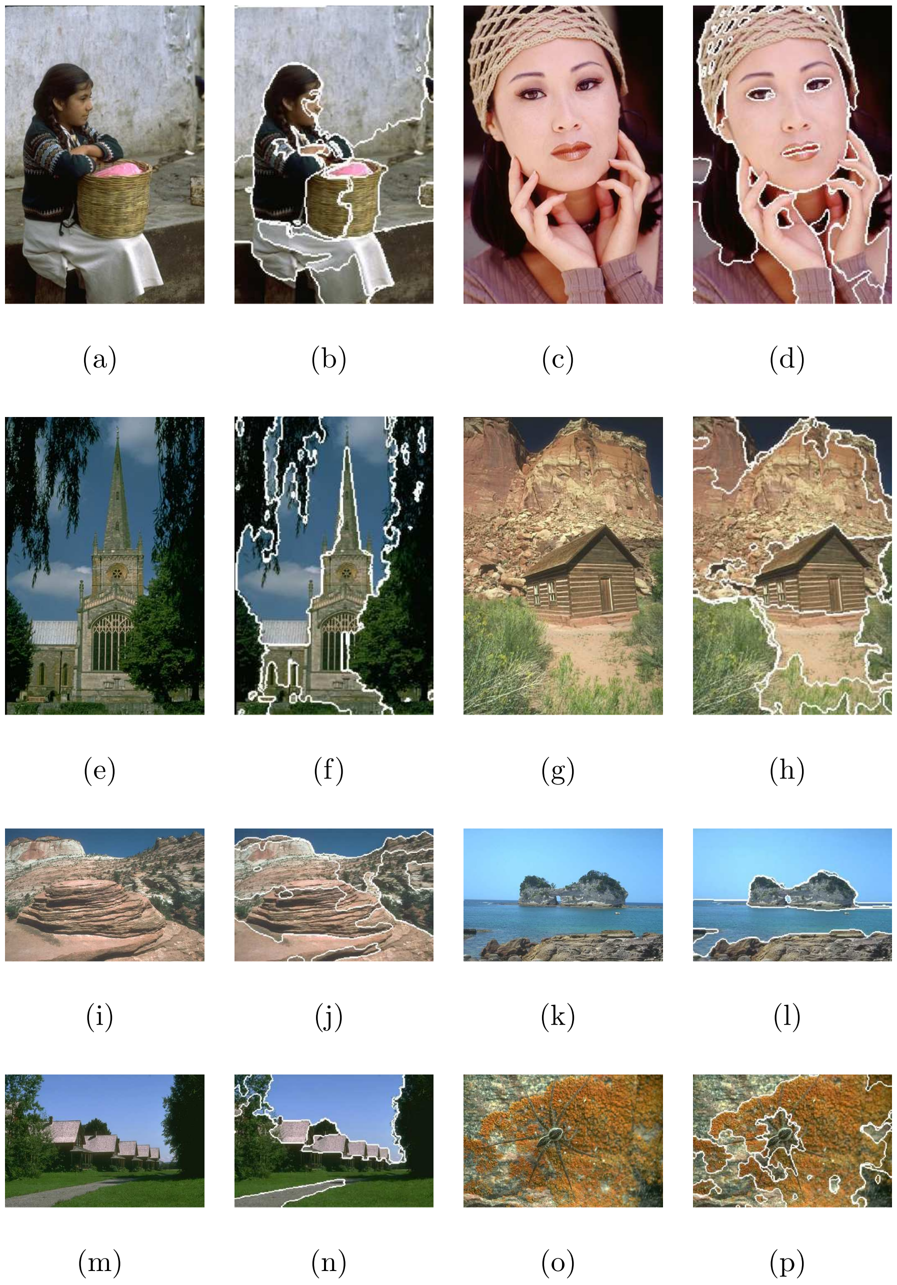
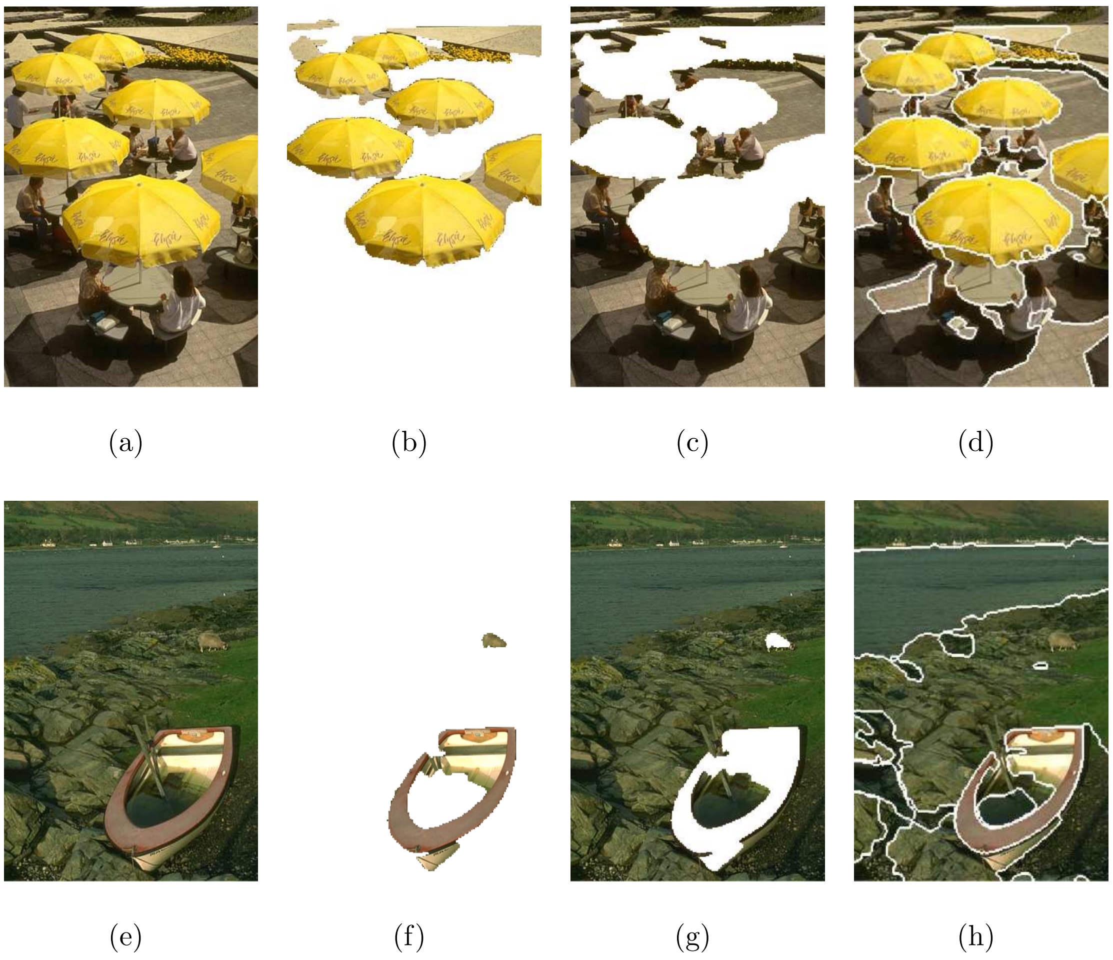


Figure 5.14: Segmentation results for the multi-region approach using coupled curve evolutions. 9 curves are initialized for each image.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.15: Comparison of background/foreground segmentation vs. multi--region segmentation using couple curve evolutions. First column shows the original images. Second and third columns correspond to foreground and background. Forth column shows the multi-region segmentation boundary.

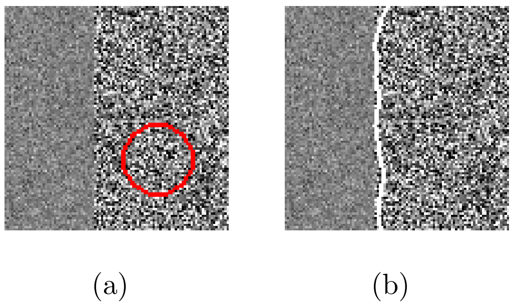


Figure 5.16: Segmentation of an image with same mean but different variations in its two regions a) Initial curve. b) Segmentation result.

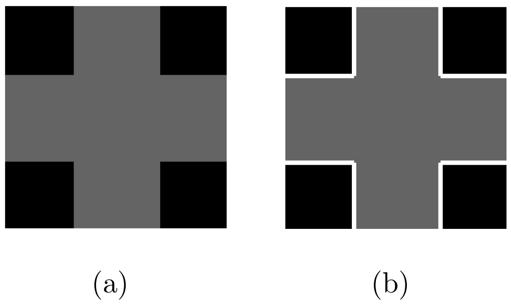


Figure 5.17: Reproduced from ratio cut paper [2]. a) Original image. b) Ratio cut segmentation after 4 bi-partitioning steps.

ibility by allowing multiple components as one region. If connected components are desired, segmentation results can be post-processed and only the component with the largest area can be selected as the partitioning.

5.12 Conclusion

In this chapter, we presented a generic region-based segmentation method for color and texture images. This method is based on defining a dissimilarity metric at the pixel level and finding the optimum partitioning of the image that maximizes the dissimilarities across the boundary. We have shown connections to GPM and derived curve evolution equations for various complex region-based segmentation cost functions. These segmentation cost functions include minimum and maximum cut frameworks and area and boundary normalized versions of these.

Region-based methods do have the tendency to fail in extracting precise boundaries. Integrating edges to the region-based framework addresses this problem. We proposed different strategies for integrating edges both from ACM and GPM point of views.

The initial theory of graph partitioning active contours is developed for bi-partitioning the image into foreground and background regions. One popular way of extending this to multi-region segmentation is by applying bi-partitioning recursively for each sub regions. This strategy works with our method very well. We also propose a novel method for multi-region segmentation based on evolving several coupled curves simultaneously.

Unfortunately, naive calculations of dissimilarities and region integrals in (5.20) bring unreasonable computational complexity and memory requirements. To address this issue, we proposed a fast method for implementing the curve evolution. This method is based on reducing the resolution of one dimension of the dissimilarity matrix W. Despite the fast implementation, we observe that precision of the segmentation is not lost. We then propose even faster implementations if some precision can be sacrificed. A multi-scale segmentation is also proposed where the precision of the segmentation is gradually increased.

An extension to interactive segmentation is also demonstrated. In this case, the user is asked to draw a rectangle covering most of the object he/she is interested in. Then, curve evolution starts within this rectangle. The curve is allowed to evolve past the boundary but the background is evaluated only within the rectangle.

We have shown promising experimental results for both bi-partitioning and multi-region segmentation. The results support the theory we developed in this paper.

One of the objectives of this chapter was to bring together ACM and GPM. Both of these techniques has shown promising and novel characteristics. We investigated and developed connections between these techniques. Hopefully this work will stimulate more research in this direction.

Chapter 6

Search and Category Pruning in Image Databases using Segmentation

With the increasing use of digital cameras, growth of the Internet and high importance of periodic acquisition of aerial and satellite images, image databases are becoming more and more important every year. The world wide web (www) itself probably contains a majority of the images in existence and these images can be assigned text captions and keywords, which makes creating image databases from web images an interesting and challenging task. Most of the image databases are or can be divided into semantic subgroups, which we call categories. One example for a category is a directory of photographs in a home user's computer. Another example of a category is the group of images in an internet image database that has the keyword “car” associated to them. Unfortunately most image databases are not generated with the semantic categorization in mind.

In this chapter, we show an efficient way to spider and simultaneously categorize web images for content based retrieval. On this database of over 600,000 images, we achieve good retrieval results by exploiting the category structure. To create this image database, we start with an existing category structure of web sites (Open Directory project - http://www.dmoz.org). We then spider images from each of these web sites and assign the images to the category from which the corresponding web site comes from. For example, the “Cars” category consists of images spidered from car related web sites. About 60-80% of images in this category are images of cars. In addition, there are also other pictures that are not of cars on such pages, e.g., picture of the owner of a car dealership. This is a common problem with automatically generated categories. To increase the overall effectiveness of retrievals, further pruning is needed. Pruning will improve both browsing and content-based retrieval quality of these databases. The main objective of the pruning is increasing the precision of the category while preserving the recall. By using our graph partitioning active contour segmentation method to separate background from foreground, we demonstrate that good image segmentation helps in this pruning step and improves the overall retrieval performance.

Image segmentation has been used in image databases as a tool to limit extraction of image features to the segments of interest. These segment-based feature vectors are then used for image or region search in a content-based image retrieval system [124, 125] and for browsing image databases [126]. In this chapter we propose a new way of utilizing segmentation for image databases. The objective of our work is to improve the quality of both image segmentation and image database retrieval simultaneously.

6.1 Category-based Database Design and Image Retrieval

Indexing diverse collections of multimedia data remains a challenging problem. Even though significant progress has been made toward developing effective content-descriptors, as evidenced by the current MPEG-7 standard, it is still difficult to bridge the gap between low-level image analysis and image understanding at the semantic level. This gap limits access solutions since users usually interact at the semantic level.

The images found on the World Wide Web (WWW) are a prime example of a multimedia collection that is difficult to index. Low-level features, such as color and texture, can be extracted and used for similarity searches. The results might be visually relevant but it is unreasonable to expect them to be conceptually relevant. For this reason the content-based searches must be constrained to semantically relevant sets of images. Due to the size of the data set, manual classification is not feasible.

This work investigates how existing semantic structure can be exploited by a multimedia access system even if this structure is not perfect. Our approach recognizes that the WWW is not just a large collection of images with loosely associated text but there is existent structure which can be exploited. The work also demonstrates how relevance feedback can refine query intent using a simple and intuitive interface.

The objective is then to index a large number of images from the WWW for efficient and accurate access given the following constraints:

- Image discovery and indexing should be automatic.
- Image search should be semantically constrained.
- The search interface should be simple and intuitive.

The proposed Categorical Image Search (CIS) system accomplishes this by:

- Using an existing directory for image discovery.
- Using the directory to constrain content-based searches.
- Using relevance feedback to learn query intent.

6.1.1 Image Ingestion

Images are collected by spidering web sites listed in the DMOZ Open Directory Project (http://www.dmoz.org). The DMOZ project's goal is to "produce the most comprehensive directory of the web, by relying on a vast army of volunteer editors." A total of 2,633,071 sites in 369,307 categories are managed by 36,786 editors at the time when this work is prepared. We developed a custom web-spider that locates images at sites parsed from a snapshot of the directory. Irrelevant images, such as icons and banners, are filtered-out. Different types of text associated with the images are stored in a relational database. This includes category names, image names, image ALT tags and HTML text. Thumbnail versions of the images are created for display purposes. Finally, texture and color content descriptors are extracted.

Texture descriptors are extracted by applying a set of Gabor-wavelet filters tuned to combinations of three scales and four orientations [123]. The feature vector components are the means and standard-deviations of the outputs of the 12 filters. The results in a 24-dimension texture feature vector. The similarity between images in the texture feature space is computed using the L_1 norm.

Color descriptors are derived from the color distribution histogram computed in the CIE L*u*v* color space. The three color dimensions are quantized to four levels resulting in a 64-dimension color feature vector. The similarity between images in the color feature space is computed using the Euclidean distance.

6.1.2 Database Organization and Image Search

The text in the relational database is compiled to create an extensive keyword list. This list is the basis of an inverted-index of the categories based on keyword frequency. The other major component of the database is the collection of image features. Each image is represented by the 88 numbers from the texture and color feature vectors. The image features are organized by category. Fig. 6.1 shows the structure of the database. Each keyword can index any number of categories but only the top 10 are retrieved during a lookup. A category can be indexed by multiple keywords.

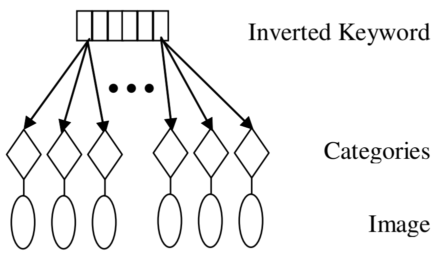


Figure 6.1: Database structure.

Image searches consists of two steps. First, a text-based search identifies the categories. Then, a constrained content-based search with relevance feedback is performed among the images from these categories.

Text Search

Image searches are initiated with keywords. The keywords determine which categories to confine the content-based search to. The inverted index is used to retrieve the image feature clusters corresponding to these categories. Constraining the content-based search improves the semantic relevance of the results and allows the search to be performed in real time.

Constrained Content-based Search

The content-based search uses a query-by-example paradigm. The user first selects one or more images from a set chosen randomly from the candidate categories. This selection is used to perform a similarity search in the combined feature spaces. If the user is not satisfied with the results, he/she can add or remove images and perform additional similarity searches. A relevance feedback loop is used to iteratively refine the query vector and search space.

Relevance Feedback

Relevance feedback is used to compute the query vector, compute the relative feature weightings, and prune the search space. This part of the search only uses the image descriptors. The keywords from the first stage of the search are no longer used and the categories only serve to constrain the search.

Query Specification:

If the user selects a single image then its 64-dimension color and 24-dimension texture feature vectors are concatenated to produce the 88-dimension query vector used in the content-based search. If multiple images are chosen then their feature vectors must be combined to produce a single 88-dimension query vector. If it is assumed that the user selects a visually similar set of images then a straightforward solution is to use the average of the vectors. In the CIS system, the 88-dimension query vector is computed to be the average of the individual vectors in the case where multiple images are chosen.

Relative Feature Weightings:

The distribution of the user-selected images is used to determine the relative feature weightings; i.e., which feature, color or texture, is more important to the user for a particular search. Intuitively, the feature space in which the images are more "tightly clustered" should be weighed more. For a set R of example images:

$$\overline{d}_{texture} = \frac{1}{|R|} \sum_{i,j \in R} d_{texture}\left(i, j\right)$$

and

$$\bar{d}_{color} = \frac{1}{|R|} \sum_{i,j \in R} d_{color}(i,j)$$

The feature weightings are then:

$$w_{texture} = \frac{1}{\bar{d}_{texture} + \varepsilon}$$

and

$$w_{color} = \frac{1}{\bar{d}_{color} + \varepsilon}$$

where ε is a small value to prevent one of the features from becoming too significant. The final distance measure used to rank-order the N nearest neighbors is:

$$d(\cdot,\cdot) = w_{texture} d_{texture}(\cdot,\cdot) + w_{color} d_{color}(\cdot,\cdot)$$

This application of relevance feedback is similar to that presented in [127].

If the user selects only one image then there is not enough knowledge to determine which feature, color or texture, is more significant. In this case, two similarity searches are performed: an N/2 nearest neighbor search using texture descriptors and an N/2 nearest neighbor using color descriptors.

Restricting Categories:

After several iterations of the relevance feedback loop, the search space is further reduced to just those categories from which the user is selecting images. This improves the relevance of the results.

6.1.3 Examples of Search and Retrieval

An implementation of the CIS system indexes over 600,000 images from 38,604 sites in 1,623 categories. It is available online in the Demos section at the web site http://vision.ece.ucsb.edu.

Figures 6.2, 6.3, and 6.4 show the different steps of a search initiated using the keyword "hat". In this case, the inverted index is used to constrain the content-based search to approximately 9,000 images in 10 categories such as Shopping:: Clothing:: Hats, Shopping:: Clothing:: Sportswear, and Shopping:: Clothing:: Costumes. From this point on, the search only uses the image-derived feature descriptors. Fig. 6.2 shows 12 images displayed at random from the constrained set. The user selects the top right image and initiates a constrained content-based search. As discussed above, the relative feature weightings cannot be determined if only a single image is selected so two nearest neighbor searches are performed. The top row of Fig. 6.3 shows the query image and the 5 most similar images with respect to the color descriptor. The bottom row shows the query image and the 5 most similar images with respect to the texture descriptor. The user adds the bottom right image and initiates another content-based search. The two selected images are used to determine the query vector and the relative feature weightings, as discussed above. The final results are shown in Fig 6.4. If necessary, the user can continue to add or remove images and perform additional content-based searches.

This example shows that the CIS system retrieves visually and conceptually relevant images within a few iterations. The integration of the semantic classification provided by the categories and the visual similarity retrieval capabilities of the image descriptors makes this possible. Neither one of these techniques alone provides relevant results. Fig. 6.2 shows that fewer than half the images chosen at random from the constrained set are related to the concept "hat". This is because the semantic classification provided by the categories is not perfect. Fig. 6.5 shows the results of an unconstrained content-based search using the same query vector and feature weightings as the search in Fig. 6.4. As expected, the concept of a "hat" is lost when the content-based search is over the entire database of 600,000 images.

Even though the nearest neighbor searches are only performed on a subset of the database, they are still computationally expensive due to the high-dimensionality of the image feature space. Conventional indexing methods, such as the R-tree and its variants, cannot be used because the similarity metric is not fixed. A novel algorithm has been developed to address this problem [128]. The method exploits the correlations between consecutive nearest neighbor searches to filter out candidate matches. This greatly reduces the number of I/O accesses in the database. The method has been shown to be effective for the 600,000 images indexed by the CIS system.

In the next section, we investigate the idea of pruning the categories and removing the outliers using image segmentation.

6.2 Category Pruning

In developing a pruning strategy using segmentation, we make the following assumptions:

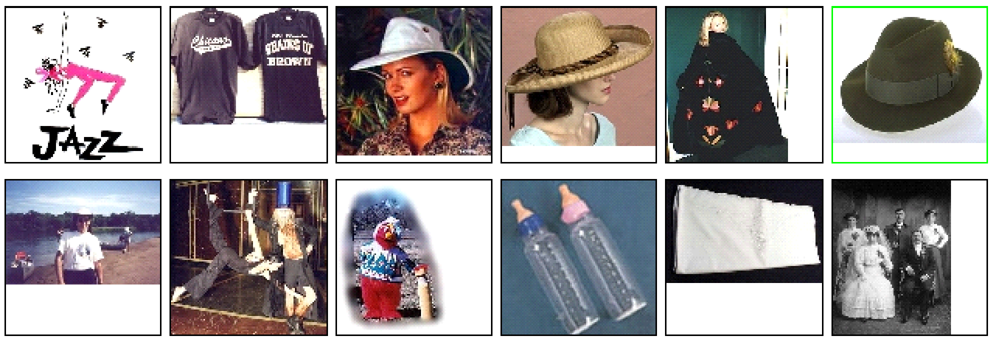


Figure 6.2: Random images from categories identified using the keyword “hat”. The top right image is selected for a constrained content-based search.

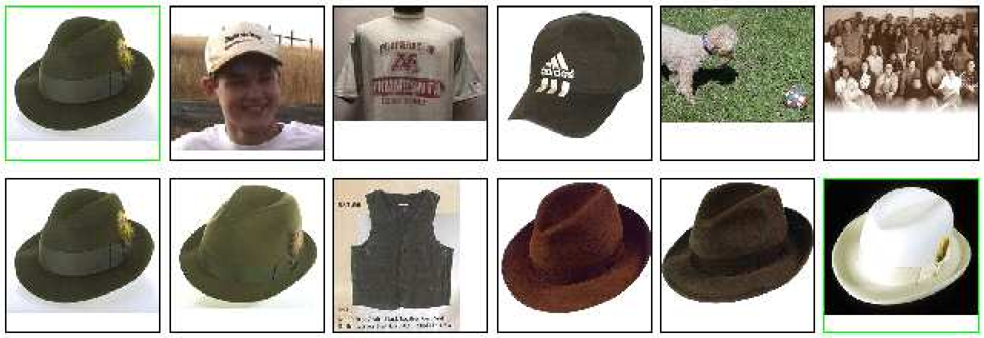


Figure 6.3: Results of a constrained content-based search using the image selected in Fig. 6.2. Top row shows query image and 5 most similar images with respect to the color descriptor. Bottom row shows query image and 5 most similar images with respect to the texture descriptor. The bottom right image is added to the query.

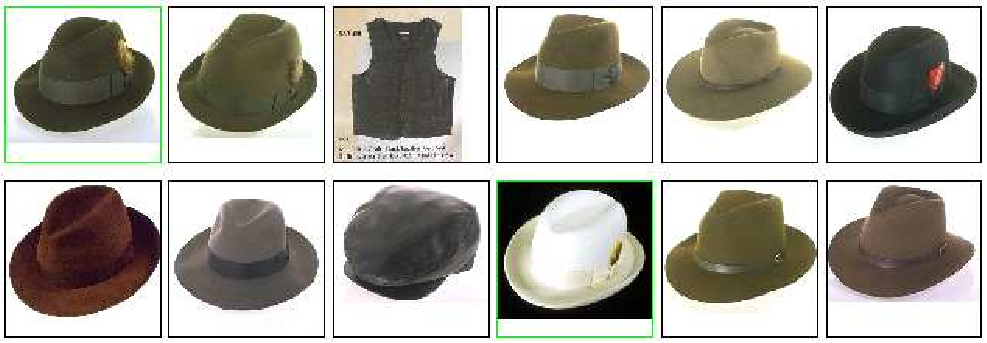


Figure 6.4: Results of constrained content-based search using the two images selected in Fig. 6.3. Relevance feedback is used to compute the query vector and relative feature weightings.

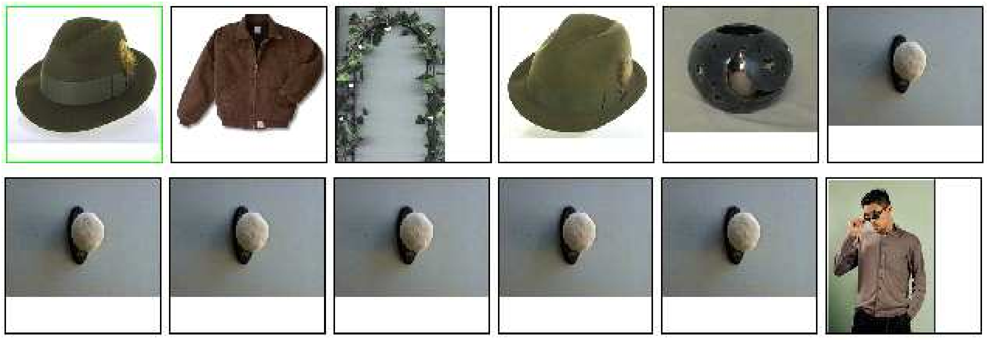


Figure 6.5: Results of an unconstrained content-based search using the same query vector and feature weightings as the search in Fig. 6.4.

- We do not have any domain specific knowledge, or object models about the category.
- The category is automatically generated, but has some consistency such that large number (more than 50%) of the images in this category fit to a certain unknown semantic concept.
- Images can vary in terms of image features or shape of the objects even if they follow the category concept.
- Images can vary in their sizes. In our case, the images are between 128×128 and 512×512.

We propose a solution to this problem by discovering a loose spatial relationship between the background and foreground of the images. The labels *background* and *foreground* have no importance in our method but are used for consistent association of regions among category images. We first segment all images into foreground and background regions using graph partitioning active contours (GPAC) (see Chapter 5). Using these segmentations, a signed distance map is calculated for each image based on the segmentation boundary. Then, these distance maps are averaged to find a distance map for the whole category. A continuous foreground/background spatial relationship within the category is captured by this average distance map. After that, the images are re-segmented, but this time the segmentation cost function also incorporates the spatial relationships discovered from the previous iteration. We show that this step improves both the segmentation of certain images and the distance map itself.

The pruning of the category is achieved by comparing the individual distance maps to the average distance map. We show the distribution of this comparison and interpret the results by showing specific examples. In addition, precision recall curves are drawn for visualization of how well our method works.

6.2.1 Segmentation Framework

We use segmentation as two-region (background and foreground) partitioning of an image as discussed in Chapter 5. We will show in Section 6.2.2 situations where multi-region (3 or 4 regions) segmentation might be necessary. Our segmentation approach, graph partitioning active contours (GPAC), is based on grouping similar points as foreground and background while increasing the dissimilarity between these regions. One important characteristic of GPAC is its flexibility in defining the segmentation cost function. By controlling the similarity measure, we can change and adapt the segmentation behavior to the problem at hand. We calculate the dissimilarities as:

$$w(p_i, p_j) = \|\vec{F}(p_i) - \vec{F}(p_j)\| + \alpha\|p_i - p_j\| \tag{6.1}$$

where $\vec{F}(p_i)$ is some low level image feature at point p_i (we use color) and the second term measures the spatial distance between two points. A spatial constraint is added to the original dissimilarity metric (5.16) used in Chapter 5. Based on this dissimilarity metric, the segmentation functional that we maximize is given by:

$$E = \iint_{R_i(C)} \iint_{R_o(C)} w(p_1, p_2) dp_1 dp_2 \tag{6.2}$$

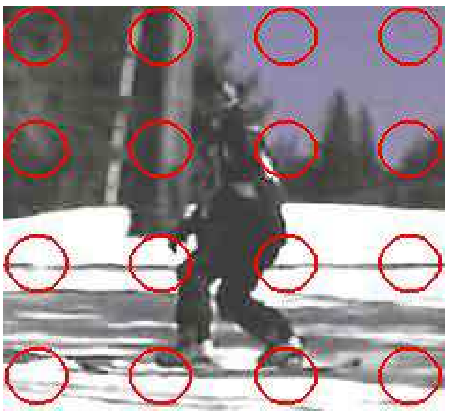


Figure 6.6: Initialization of the curve evolution for image segmentation.

where C is a closed curve on the image domain, R_i is the interior and R_o is the exterior of C. Conceptually this means finding the curve (partitioning of the image) for which the dissimilarity between its interior (foreground) and exterior (background) is maximized. This problem is solved by starting with an initial curve and maximizing (6.2) using steepest descent. After adding geometric constraints such as area normalization and boundary length minimization, the corresponding curve evolution equation becomes:

$$\frac{\partial C}{\partial t} = \left(\frac{1}{|R_o|} \iint_{R_o(C)} w(c,p)dp - \frac{1}{|R_i|} \iint_{R_i(C)} w(c,p)dp \right) \vec{N} - \gamma \kappa \vec{N} \tag{6.3}$$

This result can be visualized as the competition of foreground and background to push the curve towards the optimum boundary. Main advantage of GPAC is that its cost function is based on region features and therefore GPAC is not dependent on the initialization of the curves. Fig. 6.6 shows the curve initialization used in this paper. We initialize one multi-part curve with 16 (4×4) sub-parts, so that the curve can cover most of the image area.

6.2.2 Pruning a Category

The pruning strategy is based on first segmenting the images in a category. The signed distance of the pixels from the segmentation boundary is used as a measure for spatial relations within the images. By averaging image level spatial relations, we discover category-wide spatial relations and propose a measure for calculating the association of images to their corresponding categories.

The images in the categories are of various sizes between 128×128 and 512×512. Before segmenting these images, we resize them proportionally such that the smaller dimension of the image is mapped to 64. This small size is selected for computational reasons so that large number of images can be segmented quickly. After segmenting and extracting the boundaries, we generate a signed distance map. This map is calculated as the distance of each pixel to the segmentation boundary. The value is positive if the pixel belongs to the foreground and negative if the pixel belongs to the background. Before calculating this map, we need to decide which region is background and which region is the foreground. This is important for consistency among the images. So, we calculate the average distance from the image boundaries (not segmentation boundary) for each region. We define the foreground as the region with higher distance from the image boundaries. After calculating the signed distance map, we linearly scale the values between 0 to 255. Then using standard image resizing algorithms the distance maps are resized to the size $N \times N$. We select $N = 100$ in our experiments. The distance maps of images are averaged pixel-wise to create the average distance map of the category. We use this average map as the representation of the category.

After calculating the average map, we re-segment all images by using a new dissimilarity measure that incorporates spatial relations discovered in the previous step.

$$w(p_i, p_j) = \|\vec{F}(p_i) - \vec{F}(p_j)\| + \alpha\|p_i - p_j\| + \beta\|m(p_i) - m(p_j)\| \tag{6.4}$$

where m is the average distance map. The constant β is selected such that segmentation is not completely biased to this new term. After a second iteration of the segmentations, a new average distance map is generated. We now select this new distance map as the representation of the category's spatial relations.

To make decisions for eliminating images from the category, we need to check if an image follows the concept of the category or not. To create a measure of how well an image fits to a category, we calculate the distance between the individual distance maps and the average distance map. Distances are calculated using:

$$\|m_{avg} - m_i\| = \sqrt{\sum (m_{avg}(p) - m_i(p))^2 / N^2}$$

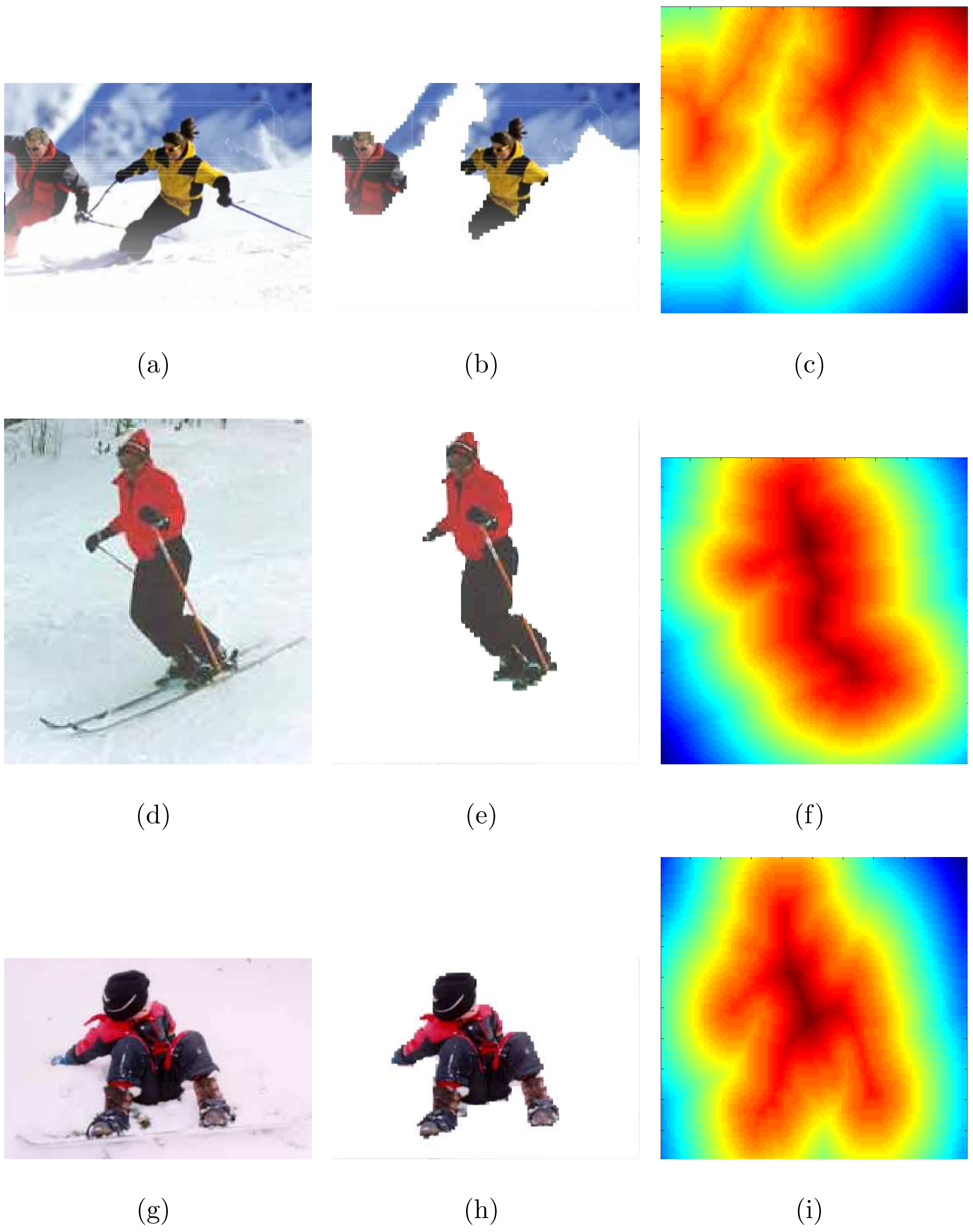
After calculating the distances, a cutoff point d_{cut} needs to be estimated, such that images with higher distance values are discarded from this category. One way is to analyze the shape of the histogram and make decisions. Simple decisions such as eliminating the tail of the histogram improves the quality of the category but this type of ad hoc decisions might be suboptimal. A better approach would be using a learning framework [129] to discover a methodology for deciding on d_{cut}. This requires generating ground truth for large number of categories.

6.2.3 Experimental Results

The category we experiment with is *Sports: Winter Sports: Skiing: Guides: North America: United States*. The main theme for the images in this category (79%–110 out of 139) is that a skier is either posing for a photograph or skiing or snowboarding downhill when the picture is taken. These images vary significantly in terms of foreground and background color, environment and pose. The rest of the images consist of scenic views of snowy mountains, maps, picture of jackets and backpacks, and a set of unrelated images. Fig 6.7 shows examples of characteristic images and Fig. 6.8 shows examples of uncharacteristic images for the category.

First, images are resized such that the smaller dimension of the image is mapped to 64. After that, all images are segmented using graph partitioning active contours. Segmentation results for some images are shown in the second column of Fig. 6.7 and Fig. 6.8. After extracting the boundaries, the signed distance maps are generated. Distance maps for some images are shown in the third column of Fig. 6.7 and Fig. 6.8. The average distance map for the category is shown in Fig. 6.9. As can be seen, the average distance map follows this category's spatial relations such that the foreground objects are in the middle part of the images in an upright position.

Using the average map, images in this category are re-segmented as discussed in Section 6.2.2. Several examples are shown in Fig. 6.10. New foregrounds in the third column show that the segmentation process has incorporated the spatial relations discovered from the category. The observation regarding the new average distance map is that this average distance map is not much different than

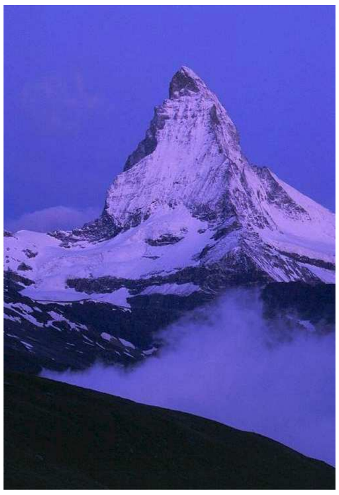


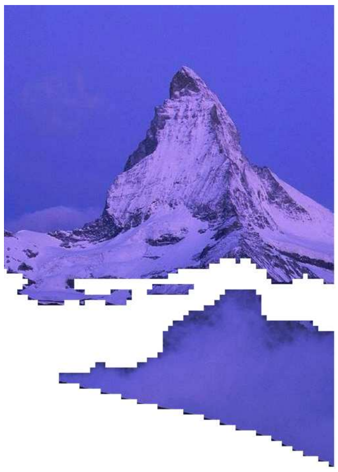
(a) (b) (c)

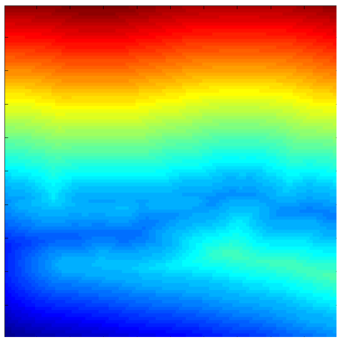
(d) (e) (f)

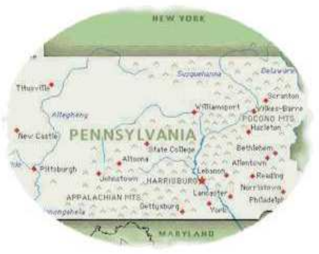
(g) (h) (i)

Figure 6.7: First column shows the original image. Foreground is shown in second column. Third column show the signed distance map.

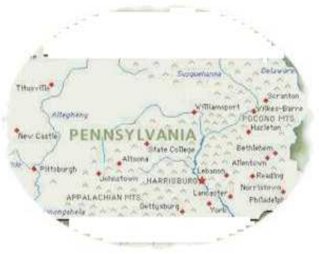

(a)


(b)


(c)



(d)



(e)

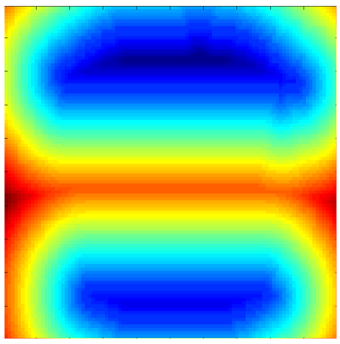
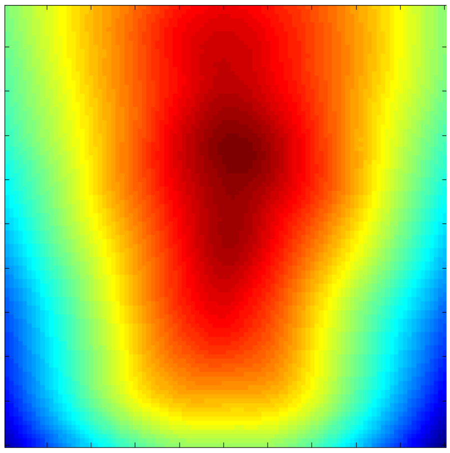

(f)

Figure 6.8: First column shows the original image. Foreground is shown in second column. Third column show the signed distance map.



(a)

Figure 6.9: Average distance map.

the previous one. The reason for this is that high percentage of the images follow the category characteristics (average distance map) and segmentations for these images did not change. This and our experiments also show that more iterations of segmentations are not necessary.

After the second iterations of the segmentations, distances of individual distance maps to the average distance map are calculated. In Fig. 6.11, the first row shows 4 images with the smallest distances and second row shows the images with the largest distance values. Fig. 6.12 (a) shows the histogram of distance values for this category. It can be seen from the figure that there are three peaks that are separated at 0.55 and 0.7. After the third peak there are outliers starting from 0.78 and up. It is also possible to use expectation maximization (EM) algorithm to fit two or three Gaussians to this data. We observe that the images within the first peak are following the category theme very well. Within the second peak,

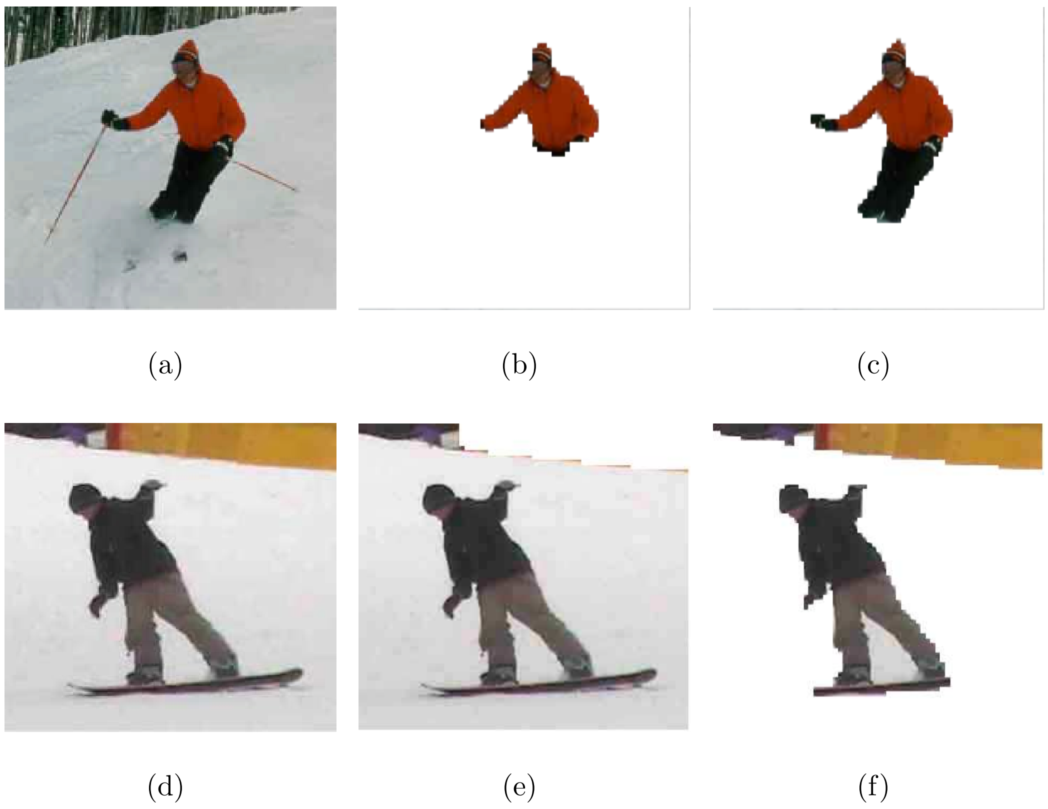


Figure 6.10: First column shows the original image. Initial foreground is shown in second column. Foreground after using spatial relations is shown in third column.

it is a combination of good and bad images for the category. Fig. 6.13 shows examples of images that fall into the second and third peak. Images in Fig. 6.13 (a-d) show that the pose of the skiers are not vertical unlike most other images. Adding some level of rotation invariance to our method might be helpful for these images. Fig. 6.13 (e-f) show two skiers using the lift. The problem with this image is that the skiers are not located in the middle part of the image while most category images have skiers in the middle parts. To handle this kind of images, our method needs to incorporate shift invariance. Last example is in Fig. 6.13 (g-h). In this example, the background consists of two different regions, the snow and the darker background corresponding to the trees. Two region segmentation is not able to handle this situation and part of the background is labeled as foreground. A 3 region or 4 region segmentation (can be achieved by recursive bi-partitioning) would be helpful when handling more complex images.

Now suppose we decide on a cutoff point d_{cut} for the distances, such that images with higher distance values are discarded from this category. Fig. 6.12(b) shows the precision-recall[1] graph for cutoff points starting with 0.25 and with increments of 0.01 up to 1.3. At 1.3, nothing is discarded, so we have 100% recall and 79% precision. We see from the graph that cutoff point 0.71 (red circle in the graph) is optimal with 96% recall and 87% precision. At this point, 5 out of 110 good images (4.5%) and 13 out of 29 bad images (45%) are discarded. We see that the precision of the category improves by 8% from 79% to 87%.

[1]After discarding images with distances bigger than a cutoff point, precision measures the percentage of good images within the remaining images in the category, and recall measures the percentage of the good images that are still left in the category.

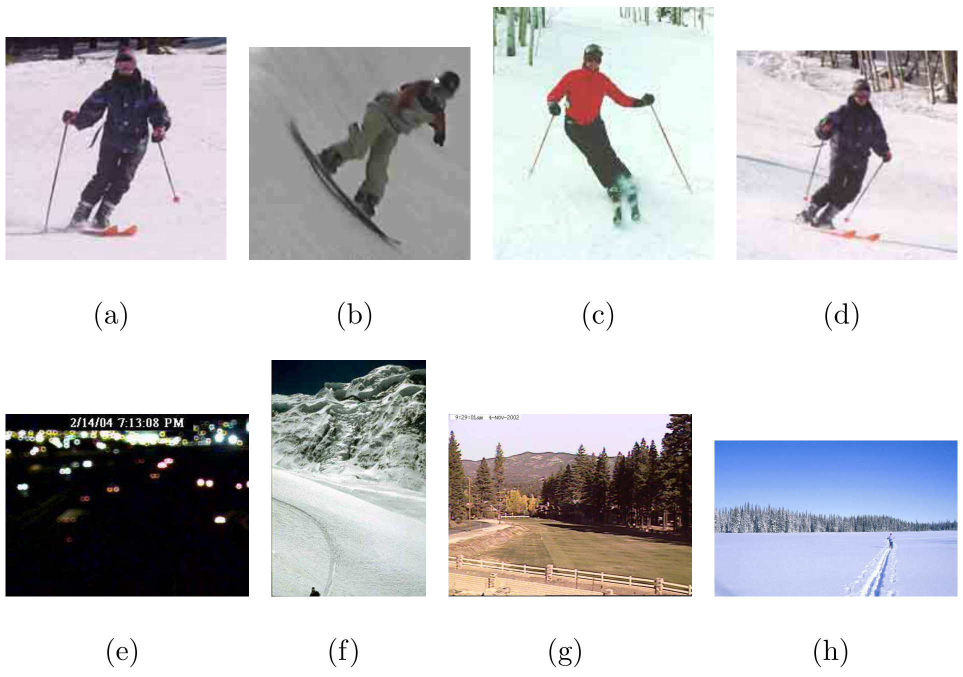


Figure 6.11: a-d) Images with smallest distances. e-h) Images with the largest distances.

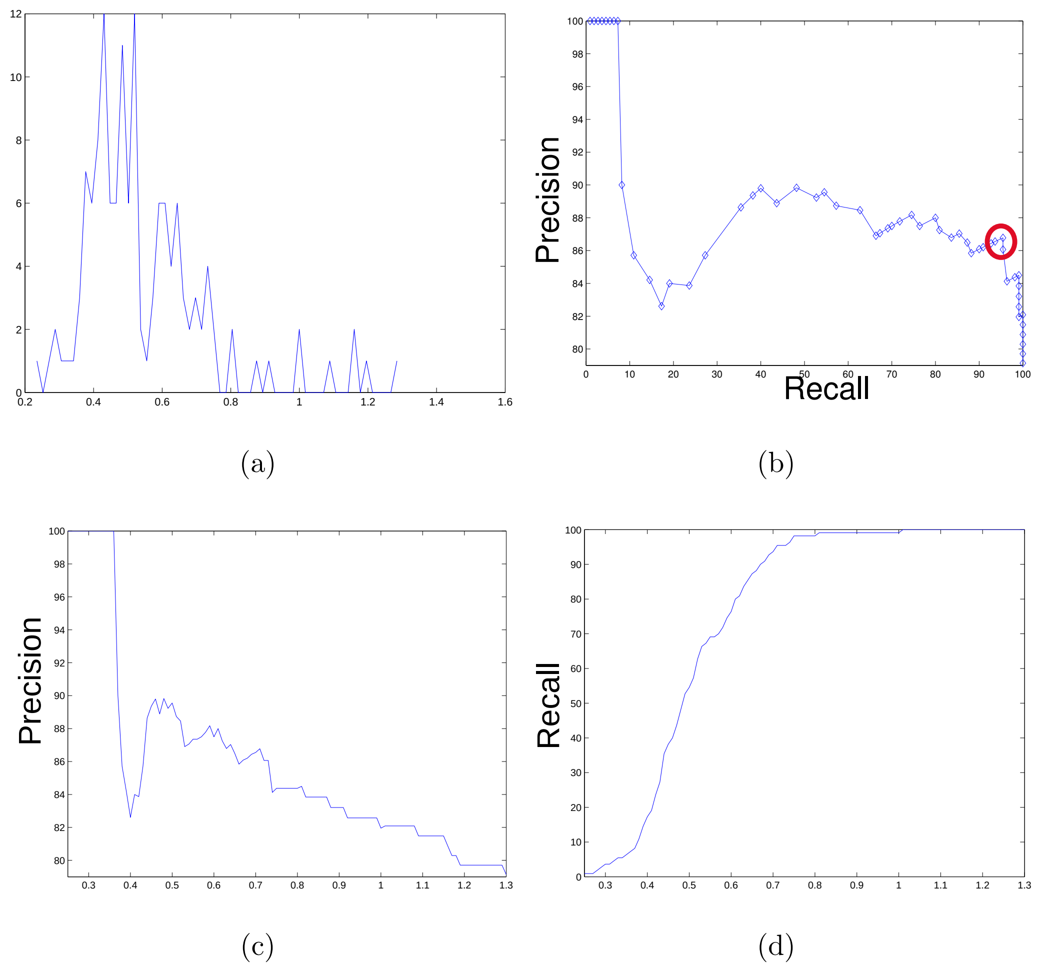


Figure 6.12: a) Histogram of the distances. b) Precision-Recall curve. c) Precision vs. cutoff point. d) Recall vs. cutoff point.

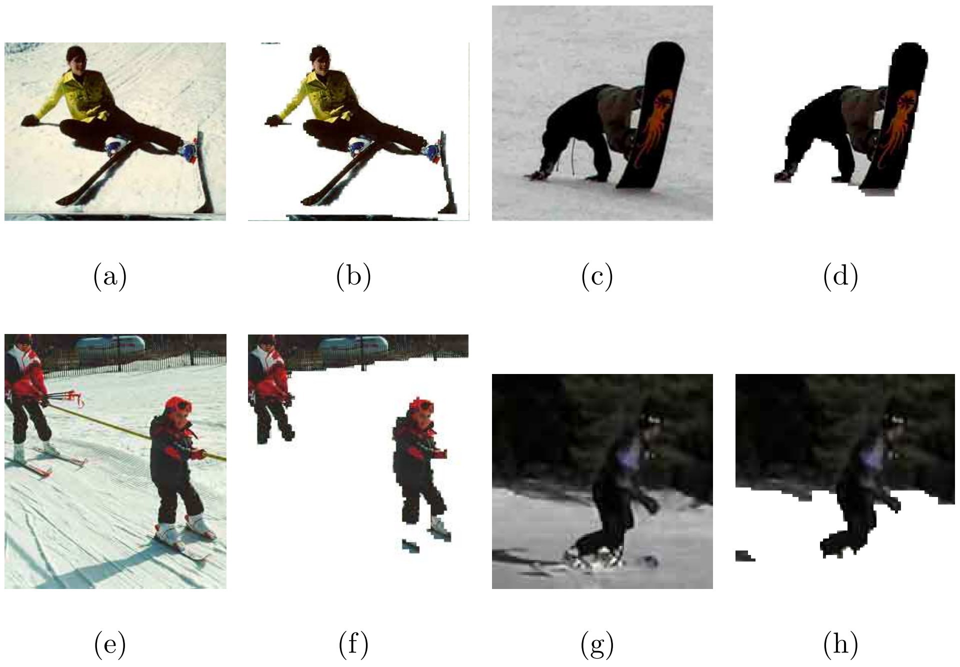

(a) (b) (c) (d)
(e) (f) (g) (h)

Figure 6.13: Images that could be misclassified.

6.2.4 Discussion

We have proposed a simple method for category pruning and discovering spatial relations in image databases. The signed distance maps created from each image can be thought of a new feature for the image. Well known image features such as color and texture can also be used together with the distance map when capturing a concept for a category. It is possible that in some categories there is no coherence within images in terms of color whereas the spatial relations are consistent. For other categories the opposite may be true. Because of the variety of images in different categories, more complex analysis of images might be necessary. Moreover, rotation invariant, scale invariant and shift invariant spatial relations or a weighted combination of these might improve pruning results for certain categories.

To further complicate the issue at hand, a category does not necessarily follow a single concept and it is likely that there are multiple groups of images in categories. Future could be to investigate a unified method that utilizes different spatial relationships and image features within a statistical learning framework to make better decisions. Other potential research directions include integration of domain knowledge (object models or shapes) into segmentation and category pruning.

Chapter 7

Future Work and Conclusion

This thesis introduced robust segmentation methods based on variational techniques. Our main goal was to design generic methods that work on natural images with little or no parameter tuning. Natural images that are rich in texture and color are among the more difficult class of images to work with, and are perhaps the most important class of images from an image understanding point of view.

Using variational methods, we investigated edge-based and region-based segmentation problems. Starting with the successful Edgeflow technique, we designed curve evolution and anisotropic diffusion techniques for image segmentation. By integrating Edgeflow technique with the variational framework, we are able to design methods that are effective on natural images and also have a sound theoretical basis.

In the second part of the thesis we investigated region-based segmentation problem again within the variational framework. Graph partitioning methods use pairwise similarity based cost functions and have been shown to work well on natural image segmentation. Starting with a similar pairwise similarity based cost function, we introduced the *maximum cut criteria* and *graph partitioning active contours* for the minimization of the corresponding cost function. Our contributions are two-fold: 1) We introduced a new type of cost function for the variational segmentation. 2) Graph partitioning methods make certain simplifications to the cost function for practical purposes and this introduces inaccuracies to the segmentation results. On the other hand, using curve evolution techniques, we are able to find the segmentation result more effectively and efficiently.

We conclude this thesis by discussing two future directions. First, we analyze the difficulties, challenges, and opportunities associated with texture edge detection. Texture edge detection has been considered a very difficult problem due to its complexity. In Section 7.1, we will discuss this problem within the Edgeflow framework. Another important future direction is associated with the image categories generated in Chapter 6. We were able to group conceptually similar images together into categories by exploiting the semantic information contained in web sites. By analyzing and mining these categories for knowledge discovery, we might be able to fine tune and further improve image segmentation. We will discuss these issues in Section 7.2.

7.1 Texture Edges

In Chapters 3 and 4 we discussed edge-based segmentation techniques, curve evolution and anisotropic diffusion, with an emphasis on brightness and color edges. On the other hand, texture is an important image feature that needs to be considered when finding the edges and object boundaries in natural images. Texture edge detection is not as well understood as brightness or color edge detection. The main reason for this is that texture is not easy to visualize or quantify. An image where each pixel corresponds to a 30 dimensional feature vector does not make much sense to an human observer.

Region-based methods have had better success with texture image segmentation. These methods, usually based on clustering the image into several regions, show reasonably good results on some images but number of regions needs to be estimated beforehand and good edge localization is not achieved in most cases. In chapter 5, we also show an example of region-based segmentation where Gabor texture features are utilized and improved the results.

One important finding in Chapter 3 was the effect of the Gaussian offset (in the calculation of Edgeflow vector field) on the final segmentation results. While original Edgeflow method [1] proposes the use of 4σ as the offset, we have shown that for gray scale and color features, using an offset of σ results in a more robust behavior. This was surprising at first. When designing the original Edgeflow vector field, Ma and Manjunath's focus was primarily on texture discrimination instead of color or gray scale features. When we conduct experiments on Gabor texture features, we observed that Gaussian offset of 4σ is a superior choice for texture edge detection when compared to σ. The reason for this behavior is that gray scale and color are point features and texture is a neighborhood feature. For example, a texture feature vector at a single pixel does not provide much information about which texture this pixel belongs to. For a small offset,

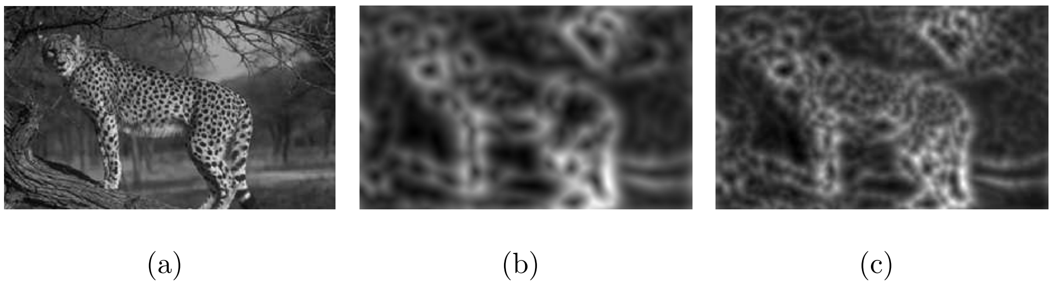


Figure 7.1: a) Cheetah image. b) Edge function generated from Edgeflow vector field with a Gaussian offset of 4σ using texture features. c) Edge function generated from Edgeflow vector field with a Gaussian offset of σ using texture features.

difference of offset Gaussians (DOOG) operator resembles derivative of Gaussian operator, which has been shown to be effective [55] for gray scale edge detection. On the other hand, at an offset of 4σ, DOOG operator can be thought of as weighted comparison of two non-overlapping adjacent neighborhoods (see Fig. 2.2). This provides an insight to why gradient-based methods had little success while Edgeflow and other neighborhood comparison methods have shown good texture discrimination on natural images. Fig. 7.1 shows a comparison of the edge functions that are generated from Edgeflow vector fields using Gaussian offsets of 4σ and σ respectively and Gabor texture features [123]. If an offset of σ is used, both edges between textures and edges within the textures are detected, which is not desired.

The original Edgeflow method compares the neighborhoods by comparing their (weighted) averages (means). As demonstrated in [123], a combination of mean and standard deviation of the texture features of image neighborhoods is a good measure for texture discrimination. We suggest that both mean and standard deviation is used when Edgeflow vector field for texture features is generated.

Edgeflow framework is not limited to comparing texture features by their means or standard deviations. Various texture dissimilarity measures ranging from histogram comparisons to information theory based dissimilarity measures can be utilized. A comparison of various dissimilarity measures for texture is given in [130]. With minor changes, Edgeflow framework can be adapted to use any of these dissimilarity measures. This also opens an opportunity for a performance comparison of various dissimilarities within Edgeflow framework.

7.2 Knowledge Discovery from Image Categories and Segmentation

The low level image processing methods we introduced within this thesis are not able to capture high level information about the scene or objects in the image. The Berkeley segmentation ground truth that we extensively utilized in Chapter 3 is one of the first efforts to "learn" some of the high level semantics and integrate these into the segmentation framework. Unfortunately, creating ground truth segmentations is very labor intensive and ground truth segmentations for about thousand images (only several hundred of them are publicly available) took more than 8 months to generate [131]. The difficulty of generating ground truth and small size of the Berkeley segmentation data set introduce limitations on the extent to which we can improve the segmentation process.

On the other hand, in Chapter 6, we have shown a way to semantically organize millions of images in an automated fashion. Visual inspection shows that these categories contain high percentage of conceptually relevant images. This automated process also introduces inaccuracies and in Chapter 6 we investigated ways to exclude the outliers from categories.

Future directions include learning from these groupings of images to improve image segmentation and pruning the categories further based on the segmentation. Most optimizations on Berkeley segmentation data set required finding single optimum values for each parameter, such as edge detection threshold, smoothing scale, etc. On the other hand, ideally these parameters should be adaptive to the image or to the type of the image. For example in Fig. 5.11, when segmenting the bear image, we needed to decide on the relative weightings of color and texture features. Finding a globally optimum weighting might not make much sense in practical applications. The relative weightings of color and texture are clearly image dependent. While for texture rich images, such as images of various animals, texture features need to be weighted more, for other types of images weighting color highly might be a better choice. Using a large image database, we might be able to find separate segmentation rules for each category. Then, given an image to be segmented, we can find its fuzzy membership to several categories and segment the image accordingly.

7.3 Conclusion

In this thesis we demonstrated that variational segmentation techniques can be successfully applied to natural images. We designed new techniques for both edge-based and region-based variational image segmentation. We have provided quantitative results comparing our proposed new methods with current state of the art algorithms. Based on these experiments, we shown that our methods significantly outperform the state of the art.

Appendix A

Euler-Lagrange of a Functional

In this section, we compute the Euler-Lagrange [98, Chapter 8] of (3.7), which leads to the edge function we use in our curve evolution. To calculate the first variation of E, suppose $\hat{V}$ is a solution of (3.7). Let K be any smooth function on U and $K = 0$ on the boundary ∂U. If we evaluate E at $V = \hat{V} + \tau K$, then $E(\tau) = E(\nabla\hat{V} + \tau\nabla K) = \int_U L(\nabla\hat{V} + \tau\nabla K)$ has a minimum at $\tau = 0$, which means $E'(0) = 0$. The first variation is:

$$E'(\tau) = \int_U \frac{\partial L(\nabla\hat{V} + \tau\nabla K)}{\partial\hat{V}_x} K_x + \frac{\partial L(\nabla\hat{V} + \tau\nabla K)}{\partial\hat{V}_y} K_y$$

Setting $\tau = 0$ and integrating by parts ($K = 0$ at ∂U) we obtain:

$$E'(0) = \int_U \left[-\partial\left(\frac{\partial L}{\partial\hat{V}_x}\right)/\partial x - \partial\left(\frac{\partial L}{\partial\hat{V}_y}\right)/\partial y \right] K = 0$$

This equation holds for all functions K, so $\hat{V}$ is also a solution of (3.8), which is the Euler-Lagrange of (3.7).

Appendix B

First Variation of a Functional

In this section we calculate the first variation for the following functional with respect to t:

$$M = \oint_C \left\langle \vec{S}, \vec{N} \right\rangle ds \tag{B.1}$$

where $\nabla \cdot \vec{S} = G$ as defined in Section 5.2. Let $\vec{S} = [S^1 \; S^2]^T$. M is then equal to:

$$M = \int_0^1 \left\langle \begin{bmatrix} S^1 \\ S^2 \end{bmatrix}, \|\vec{C}_p\| \vec{N} \right\rangle dp \tag{B.2}$$

where p is a parametrization of the closed curve $\vec{C}$. Remember that $\vec{C}_t = [x_t \; y_t]^T$, $\vec{C}_p = [x_p \; y_p]^T$ and $\|\vec{C}_p\|\vec{N} = [-y_p \; x_p]^T$. The first variation can be written as:

$$\begin{aligned} \frac{\partial M}{\partial t} &= \int_0^1 \left\langle \begin{bmatrix} \nabla S^1 \cdot \vec{C}_t \\ \nabla S^2 \cdot \vec{C}_t \end{bmatrix}, \begin{bmatrix} -y_p \\ x_p \end{bmatrix} \right\rangle dp \\ &+ \int_0^1 \left\langle \begin{bmatrix} S^1 \\ S^2 \end{bmatrix}, \begin{bmatrix} -y_{pt} \\ x_{pt} \end{bmatrix} \right\rangle dp \end{aligned}$$

Integrating by parts:

$$\begin{aligned} \frac{\partial M}{\partial t} &= \int_0^1 \left\langle \begin{bmatrix} \nabla S^1 \cdot \vec{C}_t \\ \nabla S^2 \cdot \vec{C}_t \end{bmatrix}, \begin{bmatrix} -y_p \\ x_p \end{bmatrix} \right\rangle dp \\ &- \int_0^1 \left\langle \begin{bmatrix} S^1_p \\ S^2_p \end{bmatrix}, \begin{bmatrix} -y_t \\ x_t \end{bmatrix} \right\rangle dp \end{aligned}$$

We rewrite the scalar product within the second integral:

$$\frac{\partial M}{\partial t} = \int_0^1 \left\langle \begin{bmatrix} \nabla S^1 \cdot \vec{C}_t \\ \nabla S^2 \cdot \vec{C}_t \end{bmatrix}, \begin{bmatrix} -y_p \\ x_p \end{bmatrix} \right\rangle dp$$
$$- \int_0^1 \left\langle \begin{bmatrix} \nabla S^2 \cdot \vec{C}_p \\ -\nabla S^1 \cdot \vec{C}_p \end{bmatrix}, \vec{C}_t \right\rangle dp$$

After opening the scalar products and rearranging terms:

$$\frac{\partial M}{\partial t} = \int_0^1 \left\langle \begin{bmatrix} -y_p S_x^1 + \cancel{x_p S_x^2} - \cancel{x_p S_x^2} - y_p S_y^2 \\ -\cancel{y_p S_y^1} + x_p S_y^2 + x_p S_x^1 + \cancel{y_p S_y^1} \end{bmatrix}, \vec{C}_t \right\rangle dp$$

This is equal to:

$$\frac{\partial M}{\partial t} = \int_0^1 \left\langle (S_x^1 + S_y^2) \begin{bmatrix} -y_p \\ x_p \end{bmatrix}, \vec{C}_t \right\rangle dp$$
$$= \int_0^1 \left\langle (\nabla \cdot \vec{S}) \|\vec{C}_p\| \vec{N}, \vec{C}_t \right\rangle dp$$
$$= \oint_C \left\langle G\vec{N}, \vec{C}_t \right\rangle ds$$

Bibliography

[1] Wei-Ying Ma and B S Manjunath. Edgeflow: a technique for boundary detection and image segmentation. *IEEE Transactions on Image Processing*, pages 1375–88, August 2000.

[2] S. Wang and J. M. Siskind. Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 675–690, Jun 2003.

[3] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, pages 99–121, November 2000.

[4] M.A. Ruzon and C. Tomasi. Edge, junction, and corner detection using color distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1281–1295, November 2001.

[5] A. P. Witkin. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[6] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 629–639, July 1990.

[7] Francine Catte, Pierre-Louis Lions, Jean-Michel Morel, and Tomeu Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical Analysis*, pages 182–193, February 1992.

[8] S Osher and L I Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis*, pages 919–40, August 1990.

[9] P. Saint-Marc, J.-S. Chen, and G. Medioni. Adaptive smoothing: a general tool for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 514–529, June 1991.

[10] D Mumford and J Shah. Boundary detection by minimizing functionals. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22–6, 1985.

[11] T F Chan and L A Vese. Active contours without edges. *IEEE Transactions on Image Processing*, pages 266–77, February 2001.

[12] A Tsai, A Jr Yezzi, and A S Willsky. Curve evolution implementation of the mumford-shah functional for image segmentation, denoising, interpolation, and magnification. *IEEE Transactions on Image Processing*, pages 1169–86, August 2001.

[13] Luminita A. Vese and Tony F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, pages 271–293, December 2002.

[14] A Jr Yezzi, A Tsai, and A Willsky. A statistical approach to snakes for bimodal and trimodal imagery. In *International Conference on Computer Vision (ICCV)*, pages 898–903, 1999.

[15] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 187–217, February 1980.

[16] John Canny. Finding edges and lines in images. Technical report, MIT Artificial Intelligence Laboratory AITR-720, 1983.

[17] R. Haralick. Digital step edges from zero crossings of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 58–68, January 1984.

[18] M. Concetta Morrone and D. C. Burr. Feature detection in human vision: A phase-dependent energy model. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 221–245, December 1988.

[19] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 52–57, December 1990.

[20] Stephen M. Smith and J. Michael Brady. Susan a new approach to low level image processing. *International Journal of Computer Vision*, pages 45–78, May 1997.

[21] M. Nitzberg, D. Mumford, and T. Shiota. *Filtering, Segmentation and Depth*. Springer-Verlag, 1993.

[22] M. Zuliani, C. Kenney, and B. S. Manjunath. A mathematical comparison of point detectors. In *Image and Video Registration Workshop (IVR)*, 2004.

[23] M.A. Ruzon and C. Tomasi. Color edge detection with the compass operator. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 160–166, June 1999.

[24] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 530–549, May 2004.

[25] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical roc curves. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 354–359, June 1999.

[26] M.J. Black, G. Sapiro, D.H. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, pages 421–432, March 1998.

[27] F. Heitger. Feature detection using suppression and enhancement. Technical Report TR 163, Image Schience Lab, ETH-Zurich, 1995.

[28] S. Konishi, A.L. Yuille, J. Coughlan, and Song Chun Zhu. Fundamental bounds on edge detection: an information theoretic evaluation of different edge cues. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 573–579, June 1999.

[29] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision (ICCV)*, pages 416–423, July 2001.

[30] M Kass, A Witkin, and D Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, pages 321–31, 1987.

[31] J. Shah. A common framework for curve evolution, segmentation and anisotropic diffusion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 136–142, June 1996.

[32] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[33] J A Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.* Cambridge University Press, 1999.

[34] V Caselles, R Kimmel, and G Sapiro. Geodesic active contours. *International Journal of Computer Vision*, pages 61–79, February 1997.

[35] Chenyang Xu, A Jr Yezzi, and J L Prince. On the relationship between parametric and geometric active contours. In *Asilomar Conference on Signals, Systems and Computers*, pages 483–9, 2000.

[36] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Gradient flows and geometric active contour models. In *International Conference on Computer Vision (ICCV)*, pages 810–815, June 1995.

[37] V Caselles, F Catte, T Coll, and F Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, pages 1–31, 1993.

[38] R Malladi, J A Sethian, and B C Vemuri. Evolutionary fronts for topology-independent shape modeling and recovery. In *European Conference on Computer Vision (ECCV)*, pages 3–13, 1994.

[39] L. D. Cohen. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 53(2):211–218, 1991.

[40] Song Chun Zhu and A Yuille. Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 884–900, September 1996.

[41] N Paragios and R Deriche. Geodesic active regions: a new framework to deal with frame partition problems in computer vision. *Journal of Visual Communication and Image Representation*, pages 249–68, March 2002.

[42] Bart M. Ter Haar Romeny, editor. *Geometry-Driven Diffusion in Computer Vision.* Kluwer Academic Publishers, November 1994.

[43] R. T. Whitaker and S. M. Pizer. A multi-scale approach to nonuniform diffusion. *CVGIP:Image Understanding*, 57(1):99–110, January 1993.

[44] S. Di Zenzo. A note on the gradient of a multi-image. *Computer Vision, Graphics and Image Processing*, pages 402–407, 1986.

[45] G. Sapiro and D.L. Ringach. Anisotropic diffusion of multivalued images with applications to color filtering. *IEEE Transactions on Image Processing*, pages 1582–1586, November 1996.

[46] Y Rubner and C Tomasi. Coalescing texture descriptors. In *ARPA Image Understanding Workshop*, February 1996.

[47] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel. Image selective smoothing and edge detection by nonlinear diffusion. ii. *SIAM Journal on Numerical Analysis*, pages 845–866, June 1992.

[48] L. Alvarez and L. Mazorra. Signal and image restoration by using shock filters and anisotropic diffusion. *SIAM Journal of Numerical Analyses*, 1994.

[49] Yu-Li You, M. Kaveh, Wen-Yuan Xu, and A. Tannenbaum. Analysis and design of anisotropic diffusion for image processing. In *IEEE International Conference on Image Processing (ICIP)*, pages 497–501, November 1994.

[50] G Sapiro. Vector (self) snakes: a geometric framework for color, texture, and multiscale image segmentation. In *IEEE International Conference on Image Processing (ICIP)*, pages 817–20, 1996.

[51] G Sapiro. Color snakes object segmentation. *Computer Vision and Image Understanding*, pages 247–53, November 1997.

[52] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. Fast geodesic active contours. *IEEE Transactions on Image Processing*, pages 1467–1475, October 2001.

[53] M Tabb and N Ahuja. Multiscale image segmentation by integrated edge and region detection. *IEEE Transactions on Image Processing*, pages 642–55, May 1997.

[54] L D Cohen and I Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1131–47, November 1993.

[55] J Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679–98, November 1986.

[56] Chenyang Xu and J L Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions of Image Processing*, pages 359–69, March 1998.

[57] Jianbo Shi and J Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 888–905, August 2000.

[58] S. Sarkar and P. Soundararajan. Supervised learning of large perceptual organization: graph spectral partitioning and learning automata. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 504–525, May 2000.

[59] I. H. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1075–1088, Oct 2001.

[60] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1101–1113, Nov 1993.

[61] P. Soundararajan and S. Sarkar. An in-depth study of graph partitioning measures for perceptual organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 642–660, June 2003.

[62] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 583–598, June 1991.

[63] P.T. Jackway. Gradient watersheds in morphological scale-space. *IEEE Transactions on Image Processing*, pages 913–921, June 1996.

[64] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1163–1173, December 1996.

[65] L. Shafarenko, M. Petrou, and J. Kittler. Automatic watershed segmentation of randomly textured color images. *IEEE Transactions on Image Processing*, pages 1530–1544, November 1997.

[66] K. Haris, S.N. Efstratiadis, N. Maglaveras, and A.K. Katsaggelos. Hybrid image segmentation using watersheds and fast region merging. *IEEE Transactions on Image Processing*, pages 1684–1699, December 1998.

[67] J.M. Gauch. Image segmentation and analysis via multiscale gradient watershed hierarchies. *IEEE Transactions on Image Processing*, pages 69–79, January 1999.

[68] M.W. Hansen and W.E. Higgins. Watershed-based maximum-homogeneity filtering. *IEEE Transactions on Image Processing*, pages 982–988, July 1999.

[69] A. Tremeau and P. Colantoni. Regions adjacency graph applied to color image segmentation. *IEEE Transactions on Image Processing*, pages 735–744, April 2000.

[70] I. Pitas and C.I. Cotsaces. Memory efficient propagation-based watershed and influence zone algorithms for large images. *IEEE Transactions on Image Processing*, pages 1185–1199, July 2000.

[71] Jaesang Park and J.M. Keller. Snakes on the watershed. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1201–1205, October 2001.

[72] O. Lezoray and H. Cardot. Cooperation of color pixel classification schemes and color watershed: a study for microscopic images. *IEEE Transactions on Image Processing*, pages 783–789, July 2002.

[73] Hieu Tat Nguyen, M. Worring, and R. van den Boomgaard. Watersnakes: energy-driven watershed segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 330–342, March 2003.

[74] I Vanhamel, I Pratikakis, and H Sahli. Multiscale gradient watersheds of color images. *IEEE Transactions of Image Processing*, pages 617–26, June 2003.

[75] P.R. Hill, C.N. Canagarajah, and D.R. Bull. Image segmentation using a texture gradient based watershed transform. *IEEE Transactions on Image Processing*, pages 1618–1633, December 2003.

[76] D. Dunn, W.E. Higgins, and J. Wakeley. Texture segmentation using 2-d gabor elementary functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 130–149, February 1994.

[77] B.B. Chaudhuri and N. Sarkar. Texture segmentation using fractal dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 72–77, January 1995.

[78] S.R. Yhann and T.Y. Young. Boundary localization in texture segmentation. *IEEE Transactions on Image Processing*, pages 849–856, June 1995.

[79] A. Teuner, O. Pichler, and B.J. Hosticka. Unsupervised texture segmentation of images using tuned matched gabor filters. *IEEE Transactions on Image Processing*, pages 863–870, June 1995.

[80] D. Dunn and W.E. Higgins. Optimal gabor filters for texture segmentation. *IEEE Transactions on Image Processing*, pages 947–964, July 1995.

[81] A. Laine and Jian Fan. Frame representations for texture segmentation. *IEEE Transactions on Image Processing*, pages 771–780, May 1996.

[82] P.P. Raghu and B. Yegnanarayana. Segmentation of gabor-filtered textures using deterministic relaxation. *IEEE Transactions on Image Processing*, pages 1625–1636, December 1996.

[83] S. Krishnamachari and R. Chellappa. Multiresolution gauss-markov random field models for texture segmentation. *IEEE Transactions on Image Processing*, pages 251–267, February 1997.

[84] O. Pichler, A. Teuner, and B.J. Hosticka. An unsupervised texture segmentation algorithm with feature space reduction and knowledge feedback. *IEEE Transactions on Image Processing*, pages 53–61, January 1998.

[85] T. Hofmann, J. Puzicha, and J.M. Buhmann. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 803–818, August 1998.

[86] T. Randen and J.H. Husoy. Texture segmentation using filters with optimized energy separation. *IEEE Transactions on Image Processing*, pages 571–582, April 1999.

[87] L.M. Kaplan. Extended fractal analysis for texture classification and segmentation. *IEEE Transactions on Image Processing*, pages 1572–1585, November 1999.

[88] Hsi-Chia Hsin. Texture segmentation using modulated wavelet transform. *IEEE Transactions on Image Processing*, pages 1299–1302, July 2000.

[89] M.L. Comer and E.J. Delp. The em/mpm algorithm for segmentation of textured images: analysis and further experimental results. *IEEE Transactions on Image Processing*, pages 1731–1744, October 2000.

[90] Christophe Samson, Laure Blanc-Fraud, Gilles Aubert, and Josiane Zerubia. A level set model for image classification. *International Journal of Computer Vision*, pages 187–197, December 2000.

[91] J.A. Rushing, H. Ranganath, T.H. Hinke, and S.J. Graves. Image segmentation using association rule features. *IEEE Transactions on Image Processing*, pages 558–567, May 2002.

[92] M. Acharyya, R.K. De, and M.K. Kundu. Extraction of features using m-band wavelet packet frame and their neuro-fuzzy evaluation for multitexture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1639–1644, December 2003.

[93] J.-F. Aujol, G. Aubert, and L. Blanc-Feraud. Wavelet-based level set evolution for classification of textured images. *IEEE Transactions on Image Processing*, pages 1634–1641, December 2003.

[94] P Brodatz. *Textures: A photographic album for artists and designers*. Dover, 1966.

[95] Zhuowen Tu, Song-Chun Zhu, and Heung-Yeung Shum. Image segmentation by data driven markov chain monte carlo. In *IEEE International Conference on Computer Vision (ICCV)*, pages 131–138, July 2001.

[96] Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 657–673, May 2002.

[97] Xiaofeng Ren and J. Malik. Learning a classification model for segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 10–17, October 2003.

[98] Lawrence C Evans. *Partial Differential Equations*. American Mathematical Society, June 1998.

[99] Do Hyun Chung and G. Sapiro. On the level lines and geometry of vector-valued images. *IEEE Signal Processing Letters*, pages 241–243, September 2000.

[100] H. Tek and B.B. Kimia. Image segmentation by reaction-diffusion bubbles. In *International Conference on Computer Vision (ICCV)*, pages 156–162, June 1995.

[101] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

[102] N. Paragios, O. Mellina-Gottardo, and V. Ramesh. Gradient vector flow fast geometric active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 402–407, March 2004.

[103] C Xu and JL Prince. Generalized gradient vector flow external forces for active contours. *Signal Processing*, pages 131–9, December 1998.

[104] A Rosenfeld and M Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, pages 562–9, May 1971.

[105] Fredrik Bergholm. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–741, 1987.

[106] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, pages 117–156, November 1998.

[107] J.H. Elder and S.W. Zucker. Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 699–716, July 1998.

[108] D.H. Marimont and Y. Rubner. A probabilistic framework for edge detection and scale selection. In *International Conference on Computer Vision (ICCV)*, pages 207–214, January 1998.

[109] Christine Bredfeldt and Dario Ringach. Dynamics of spatial frequency tuning of macaque v1. *The Journal of Neuroscience*, pages 1976–1984, March 2002.

[110] Gnther Wyszecki and W. S. Stiles. *Color Science : Concepts and Methods, Quantitative Data and Formulae.* Wiley-Interscience, 1982.

[111] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1222–1239, Nov 2001.

[112] I Kovacs and B Julesz. A closed curve is much more than an incomplete one: effect of closure in figure-ground segmentation. In *Proc Natl Acad Sci USA*, pages 7495–7, August 1993.

[113] James Elder and Steven Zucker. The effect of contour closure on the rapid discrimination of two-dimensional shapes. *Vision Research*, pages 981–991, May 1993.

[114] K Koffka. *Principles of Gestalt Psychology.* Harcourt, 1967.

[115] B Sumengen, B S Manjunath, and C Kenney. Image segmentation using curve evolution and flow fields. In *IEEE International Conference on Image Processing (ICIP)*, pages 105–8, 2002.

[116] Andy Tsai. *Curve Evolution and Estimation-Theoretic Techniques for Image Processing.* PhD thesis, Harvard-MIT Division of Health Sciences and Technology, August 2000.

[117] B Sumengen, B S Manjunath, and C Kenney. Image segmentation using curve evolution and region stability. In *International Conference on Pattern Recognition (ICPR)*, pages 965–8, August 2002.

[118] I. J. Cox, S. B. Rao, and Yu Zhong. Ratio regions: a technique for image segmentation. In *International Conference on Pattern Recognition (ICPR)*, pages 557–564, August 1996.

[119] A. Vasilevskiy and K. Siddiqi. Flux maximizing geometric flows. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7–14, July 2001.

[120] Guillermo Sapiro. *Geometric Partial Differential Equations and Image Analysis.* Cambridge University Press, January 2001.

[121] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: cue integration in image segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 918–925, September 1999.

[122] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, pages 7–27, June 2001.

[123] B S Manjunath and W Y Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 837–42, August 1996.

[124] W. Y. Ma and B. S. Manjunath. Netra: a toolbox for navigating large image databases. In *International Conference on Image Processing (ICIP)*, pages 568–571, October 1997.

[125] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1026–1038, August 2002.

[126] B. S. Manjunath and W. Y. Ma. Browsing large satellite and aerial photographs. In *International Conference on Image Processing (ICIP)*, pages 765–768, September 1996.

[127] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 644–55, September 1998.

[128] Peng Wu. *Search and indexing of high demensional feature spaces for similarity retrieval.* PhD thesis, UC, Santa Barbara, 2001.

[129] Vladimir Cherkassky and Filip Mulier. *Learning from Data: Concepts, Theory, and Methods.* Wiley-Interscience, March 1998.

[130] J. Puzicha, J.M. Buhmann, Y. Rubner, and C. Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *IEEE International Conference on Computer Vision*, pages 1165–1172, September 1999.

[131] David Royal Martin. *An Empirical Approach to Grouping and Segmentation.* PhD thesis, University of California, Berkeley, 2002.