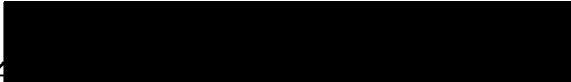


## AN ABSTRACT OF THE THESIS OF

Jin Jia for the degree of Master of Science in Computer Science presented on April 12, 2004.

Title: Object Highlighting: Real-Time Boundary Detection using a Bayesian Network.

Abstract approved: 

Eric Mortensen

Image segmentation continues to be a fundamental problem in computer vision and image understanding. In this thesis, we present a Bayesian network that we use for object boundary detection in which the MPE (most probable explanation) before any evidence can produce multiple non-overlapping, non-self-intersecting closed contours and the MPE with evidence—where one or more connected boundary points are provided—produces a single non-self-intersecting, closed contour that accurately defines an object's boundary. We also present a near-linear-time algorithm that determines the MPE by computing the minimum-path spanning tree of a weighted, planar graph and finding the excluded edge (i.e., an edge not in the spanning tree) that forms the most probable loop. This efficient algorithm allows for real-time feedback in an interactive environment in which every mouse movement produces a recomputation of the MPE based on the new evidence (i.e., the new cursor position) and displays the corresponding closed loop. We call this interface “object highlighting” since the boundary of various objects and sub-objects appear and disappear as the mouse cursor moves around within an image.

©Copyright by Jin Jia

April 12, 2004

All Rights Reserved

Object Highlighting: Real-Time Boundary Detection using a Bayesian Network

by

Jin Jia

A THESIS

submitted to

Oregon State University

in partial fulfillment of  
the requirements for the  
degree of

Master of Science


Presented April 12, 2004

Commencement June 2004

Master of Science thesis of Jin Jia presented on April 12, 2004.


APPROVED:

  
Major Professor, representing Computer Science

  
Associate Director of the School of Electrical Engineering & Computer Science

  
Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

  
Jin Jia, Author

## ACKNOWLEDGEMENTS

I would first and foremost like to express my gratitude towards my advisor, Eric Mortensen, for his support, encouragement and guidance. I would also like to thank him for introducing me to an area of Computer Science that I have come to love, but had not had much experience with prior to meeting and working with him.

I also would like to thanks Professor Bruce D'Ambrosio for his wonderful class and helpful discussion about Bayesian network. Thanks Xin Wang for her help during class.

I would like to thank Professor Tim Budd for his help during admission and my internship. I would like to thank Professor Ron Metoyer, Professor Prasad Tadepalli and Professor David Birkes for being on my committee.

Thanks, also, to all members of our research lab, but in particular to those who helped me out and discussed ideas with me, like Silvio, Hongli, Wei and Ledah. I feel proud to work with these good people. I also extend this thanks to everyone else in Dearborn 222, who made working in the lab more enjoyable. I would like to thank Shriprakash for helping me checking my thesis.

Finally, I would like to thank my parents, to whom this thesis is dedicated, for their understanding and constant support.

# TABLE OF CONTENTS

	<u>Page</u>
<b>1. INTRODUCTION.....</b>	<b>1</b>
1.1 Introduction .....	1
1.2 Statement of the Problem .....	5
1.3 Format and Chapter Overview .....	6
<b>2. BACKGROUND AND RELATED WORK .....</b>	<b>7</b>
2.1 Optimization in Object Boundary Detection.....	7
2.1.1 Region Based Optimization.....	8
2.1.2 Low Level Cues Based Grouping.....	9
2.1.3 Model Based Approach .....	13
2.2. Bayesian Network in Computer Vision .....	14
2.2.1 Bayesian Network in Image Interpretation.....	14
2.2.2 Bayesian Network in Object Recognition and Segmentation .....	16
2.3. Intelligent Scissors .....	17
<b>3. OBJECT HIGHLIGHTING .....</b>	<b>20</b>
3.1 Image Data Preprocessing.....	22
3.1.1 Tobogganing.....	22
3.1.2 Weighted graph creation.....	23
3.1.3 Computing Edge Confidence .....	25
3.2 Bayesian Network Overview .....	26
3.3 Our Bayesian Network Model.....	27
3.3.1 The Two Layer Topology.....	30
3.3.2 Probability Tables.....	32

## TABLE OF CONTENTS (Continued)

3.4	Bayesian Inference .....	39
3.4.1	MPE in Our Bayesian Network.....	40
3.4.2	MPE without Evidence.....	41
3.4.3	Single Loop MPE with Evidence .....	43
3.5	Graph Search and Loop Detection .....	47
3.6	Object Highlighting Interface.....	51
4.	RESULTS AND DISCUSSION.....	54
4.1	Single Observation .....	55
4.2	Multiple Observations.....	60
5.	CONCLUSION AND FUTURE WORK .....	66
5.1	Conclusion.....	66
5.2	Future Work .....	67
	BIBLIOGRAPHY.....	69

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
3.1 An overview of the entire Object Highlighting process. ....	22
3.2: Examples of Tobogganing and weighted graph creation. ....	24
3.3 shows the edge confidence map for the image in Figure 3.2 (a). ....	26
3.4: Creation of our Bayesian network. ....	29
3.5: Two layer graph topology example in Netica format. ....	31
3.6: Table for E-Node. ....	34
3.7: Table for V-node ....	36
3.8: An example of MPE without evidence on a 128 by 128 image. ....	43
3.9: Loop detection using a spanning tree. ....	47
3.10: The algorithm of single loop MPE with evidence. ....	50
3.11: Plot showing time to compute the minimum-path spanning tree. ....	51
3.12: Object Highlighting on the object of figures 3.4 and 3.10. ....	52
4.1 Grayscale pocket knife boundary defined with single observations. ....	57
4.2 Grayscale test images ....	57
4.3 Grayscale block and hat images with single observation. ....	58
4.4: Grayscale image of stones on pavement. ....	59
4.5: Color still life image. ....	59
4.6: Color cow boundary defined with three observations. ....	62
4.7: Grayscale spoon boundary defined with three observations ....	63
4.8: Grayscale egg boundary defined with two observations. ....	64
4.9 Color human face boundary defined with two observations. ....	65
4.10 Color sheep boundary define with thre observations. ....	65



# Object Highlighting: Real-Time Boundary Detection using a Bayesian Network

## 1. Introduction

### 1.1 Introduction

*"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightness and nuances of color. Do I have "327"? No. I have sky, house, and trees." --Max Wertheimer (1923)*

What is an image? A matrix of numbers or a square of colored dots? When we look at an image, we usually do not perceive it as individual dots or numbers; instead, we see a representation of scene. But to a computer, an image is just a collection of numbers. How does the computer covert these numbers to a scene description? The goal of computer vision is to have the computer understand images and video. This problem turns out to be enormously difficult for artificial intelligence and computer vision.

Image understanding tries to grasp the meaning of images by converting the pixel values to a scene description. Image segmentation, which partitions an image into sets of pixels that correspond to “objects”, is a key step in image understanding. The first question we ask is: how human visual perception system does object segmentation? This problem is called visual grouping and was first studied in Psychology by the Gestalt school of visual perception nearly a century ago, which was led by Max Wertheimer. This problem is also called visual grouping. The school claimed that the mind perceives the world as objects, as wholes, not as atomic

primitives. This is the Gestalt laws of perceptual organization, which defines the visual cues for perception, such as luminance similarity, proximity, continuity, connectivity, symmetry, convexity, closure and collinearity. The summarized visual cues are used by humans to do image segmentation.

However, the Gestalt laws only provide some basic principles for grouping, not a detailed theory of human perception. Certainly, there is a much more complicated process going on within the human visual system other than the simple principles provided by Gestalt Laws. For example, it may be that humans recognize objects because they have a database of those objects in some kind of compressed or abstract form. They remember those objects and even the related context. Even during occlusion, humans recognize an object just by seeing part of it. In this case, humans may use object recognition to help with segmentation or do both at the same time. How the human visual system really does object recognition and segmentation is still like a black box to us. Even though the Gestalt laws are far from a detailed theory, they do provide powerful principles that we can use. Currently, most visual grouping research is based on the Gestalt laws.

Currently, most of the research in image segmentation is focusing on fully automatic segmentation [34]. Complete automation is certainly preferred for such tasks as robotic navigation, image and video compression, model driven object delineation, multiple image correspondence, image-based modeling, or anytime autonomous interpretation of image and video is desired. The primary goal of many image segmentation techniques is to accurately find object boundaries in a fully

automatic fashion. Despite the large amount of research, no fully automatic technique currently exists that correctly segments objects in a general class of images. Due to the lack of understanding of how human visual perception deals with the complexity and tradeoff among different situations, it is unlikely that the goal of accurate and fully automatic segmentation will be achieved in the near future.

General purpose image/video editing and analysis will continue to require human guidance [35,36]. This is due to three reasons. First, as mentioned, all current fully automatic methods sometimes fail and produce incorrect results. Second, most current advanced automatic segmentation algorithms are not real time, which take several minutes and even hours to run. Third, image/video editing and analysis are creative processes and subjective problems. Humans play the essential role and identify which image components, objects or subobjects, are of interest. As we know, the user knows exactly what he wants and where the correct object boundaries should be. One method is to let the user interactively select the correct boundary from a collection of good solutions given by the computer. Compared to manually tracing the object boundary, this approach saves users' low level tedious work. This approach is called semi automatic or interactive image segmentation. Compared to the fully automatic approach, there is relatively much less research in this area. This interactive approach has successful applications in areas like image composition and medical image analysis. The goal of researches in this area are to minimize the human effort and hopefully finally achieve one step object selection, which means the right object boundary is highlighted instantly after user point to that object.

Regardless of different algorithms or tools being developed, there are some properties that are desirable in interactive image segmentation [34]. These properties include:

- *Simple*: while the underlying computation may be complex, the default interface presented to the user should be straightforward.
- *Fast*: User need instant feedback and does not want to wait between mouse clicks. It is usually desirable to be real time.
- *Intelligent*: it “understand” what the user wants based on a minimum of input- the less effort and time a user needs to direct the tool to achieve the desired result, the more intelligent the solution.
- *General*: it has broad applicability and can be used effectively in a wide variety of situations.
- *Robust*: it is forgiving, to a degree, of noise in the input, both image and human. It accepts rough or approximate inputs and noisy images and still produces the desired results.

It is not necessary for a tool to possess all of these properties to be effective. Conversely, some properties have trade off and could necessitate others. For example, the more intelligent the interactive segmentation method is, the more the need for computation time, which could slow down user interactions.

## 1.2 Statement of the Problem

Boundary detection in digital images continues to be one of the fundamental problems for computer vision and image understanding applications. Despite the large amount of research, no fully automatic technique currently exists that correctly identifies objects of interest in a general class of images. General image segmentation will continue to require some amount of human intervention. As such, our research approach is to reduce the human effort required for accurate and reliable object definition by exploiting high-level visual expertise [34,38].

Intelligent Scissors has proven to be an effective general interactive tool [37,38]. Compared to traditional low level segmentation techniques, it is fast, robust and requires relatively little human guidance to define an object boundary. The aim of this research is adding more intelligence to Intelligent Scissors framework, which could reduce human effort even more.

In this thesis, we are using Bayesian networks, which is one graphical modeling technique in artificial intelligence that has been drawing lots of attention recently [3,20]. We present a two-layer Bayesian network based on the pre-segmented image graph. It captures the special relationships of edges and nodes in the feature graph. The probability tables of this network enforce simple, non-self-intersecting closed contours after some observation is provided in the form of a sequence of connected contour points.

Exact inference in Bayesian network is generally NP-hard [5]. But with our formulation on the probability tables, the exact MPE can be determined in a fraction of

a second using a near linear-time graph search algorithm. As such, we are able to dynamically compute the MPE in an interactive environment we call *Object Highlighting*. As the user moves the mouse, each new mouse position provides a new observation in the form of a point on an object boundary. Based on the new evidence, our graph search algorithm then computes the MPE by generating a bi-directional shortest path tree. Branches of the tree form closed loops, which are the possible object boundaries.

### **1.3 Format and Chapter Overview**

The main problem addressed in this thesis is real time object boundary detection. The organization of this thesis is as follows.

Chapter 1 is the introduction. Chapter 2 presents various recent optimization techniques in image segmentation and different applications of Bayesian network in computer vision. It also discusses the advantages and possible improvements to the Intelligent Scissors framework. Chapter 3 describes the Object Highlighting approach and our Bayesian network formulation in detail. Chapter 4 demonstrates the utility of our approach by showing the results of different images. The image segmentation or boundary detection experiments are carried out in various challenge situations. Finally, Chapter 5 concludes with a discussion of future directions.

## **2. Background and Related Work**

There is a large amount of literature on segmentation dating back over 30 years. In the following, I will give an overview of recent work in object boundary detection and the applications of Bayesian network in computer vision. These methods can be roughly divided into four kinds: region based optimization, low level cue based grouping, model based approach and Bayesian networks in computer vision. In particular, I will first review all above techniques except those related to Bayesian networks. Then, there will be a discussion about recent applications of Bayesian networks in computer vision. After that, I will examine Intelligent Scissors, including its advantages and possible improvement.

### **2.1 Optimization in Object Boundary Detection**

Most recently developed techniques for object boundary detection have one thing in common: optimization. It obtains global characteristics based on the accumulation of local features. Different methods for global optimization are reviewed in this section: region based optimization, low level cues based grouping and model based approach. Actually, some methods, like Leung and Malik's work that discussed later [31], mix different approaches. Another thing in common to all the methods is using some cues in Gestalt laws of perceptual organization, but using them in different ways.

### *2.1.1 Region Based Optimization*

Region based grouping or segmentation methods are probably less common than edge detection as a low level processing operation in computer vision systems. On the one hand, they are applicable in environments that are highly textured or colored, for example outdoor scenery viewed by a mobile vehicle. They are robust to noise and tend to be global. On the other hand, they often ignore important boundary properties such as smoothness.

There are two approaches: top-down region partition and bottom up region merging. Both use graphs to represent the data. Graph based methods consider a graph whose nodes are the image features and whose edges are weighted according to some measure of similarity between nodes.

There is a class of segmentation methods based on finding an optimal solution in a graph by minimizing the similarity between pixels. For the bottom up approach, Felzenszwalb and Huttenlocher [15] recently developed a near linear time algorithm for image segmentation. The method is based on evidence for a boundary between each pair of neighboring regions, which is actually comparing the similarity of neighboring region pixels. The optimization is done by their algorithm, which is a variation of Kruskal's algorithm for constructing the minimum spanning tree (MST) of the graph.

For the top-down approach, Z. Wu and R. Leahy [57] use minimum cut for image segmentation. They formulate data clustering as minimum graph cut. Clustering is achieved by removing edges of image graphs to form mutually exclusive subgraph such that the largest inter-subgraph maximum flow is minimized. The method is



finding minimum cuts in a graph, where the cut criterion is designed in order to minimize the similarity between pixels. The disadvantage is that this method is biased towards finding small components.

Shi and Malik [48] developed a normalized cut technique, which provides a significant advance over previous work. This technique uses a generalized eigenvalue method to find normalized cuts of an image graph, and uses it to partition the image by iterating the algorithm. Cut is a measure of association. Minimizing it will give a partition with the maximum disassociation. The method measures similarity between the two groups, normalized by the “volume” they occupy in the whole graph. However, the normalized cut also yields NP-hard computational problem. Shi and Malik developed an approximation methods but the error of this approximation is not well understood. Even the approximation is fairly computationally expensive, which requires several minutes for a general image.

Leung and Malik [31] extend the normalized cut by incorporating weak contour continuity information into the region-based model. They use orientation energy to get the soft contour information. They also use contour propagation to complete the weak contrast gaps and subject contours.

### *2.1.2 Low Level Cues Based Grouping*

Low level cues based grouping is the most widely used approach for image segmentation in computer vision recently. The low level cues are included: similarity of pixel brightness, proximity, continuity and texture. After detecting the cues,

different optimization methods are used to combine the information of the cues associated with boundaries. For example, some current methods do some kinds of local edge detection, by Canny or other ways [4], then groups the edges together.

Saliency measure is one of the most important methods of perceptual grouping of edges. This approach incorporates the local laws of proximity of edges and smooth-continuation of contours in some form or other. Successive edges of a contour are expected to be in close proximity and piece-wise smooth.

Parent and Zucker [42] present an approach for curve inference that is based on curvature information. They use the concept of co-circularity as a constraint to assign tangent and curvature to every point in the image. To aid the process of curve inference, relaxation labeling is used to highlight image elements, like tangent and curvature, which are consistent with its neighbors.

Shashua and Ullman [47] present a parallel network model for detecting salient boundaries based on fragment proximity, boundary length and boundary smoothness. After rating the best connections between each edge and its neighbors, a “saliency network” of locally connected elements is created. The research done by Guy and Medioni [22] used a voting scheme to estimate a tangent and a salience at each pixel in the image.

In previous algorithms, we have seen that different visual cues from local Gestalt laws have been applied to organize local edges into extended contours, which include luminance similarity, proximity, continuity and smoothness constraints. The closure constraints have been largely ignored. There are persuasive psychophysical

demonstrations that contour closure is a stronger global constraints compared to other cues in perceptual organization [12,18]. The following researches incorporate closed contours in segmentation of natural images.

As a recent significant advance over previous works, Mahamud and Williams [33,40] show a method that reliably segments object boundaries using a saliency measure based on the global property of contour closure.

In this paper, they derive an explicit relationship between the saliency measure, which is the relative likelihood that smooth closed contours pass through a given edge, and the corresponding components of the eigensolution. Then they present a segmentation algorithm that utilizes the saliency measure to extract out multiple closed contours by finding strongly-connected components on an induced graph. This approach reduces the time taken to segment real images from an average of around 2 hours to about 10 seconds per object on a general-purpose workstation. Besides improving efficiency, this method works for object in background with natural textures. There are still some limitations in all the saliency measure, which are from its convex and smoothness assumption about object contour.

Elder and Zucker [12] develop a probabilistic model for object boundary detection in which maximum-likelihood contours are computed using a shortest path algorithm. It guarantees an exact solution in polynomial time, but favors the shortest loop and lacks global completeness and non-self-intersection constraints.

Eric Saund [46] presents an effective algorithm especially suited for finding perceptually salient compact closed region structure in hand-drawn sketches on

whiteboards and contours from photographic imagery. They start with a graph of curvilinear fragments whose proximal endpoints form junctions. The key problem is to manage the search of possible path continuations through junctions in an effort to find paths satisfying global criteria for closure and figural salience. Firstly, best-first bidirectional search is used to check for the cleanest, most obvious paths. Then apply more exhaustive search to find paths cluttered by blind alleys.

Jermyn and Ishikawa [26] describe a new form of energy functional for modeling and identification of regions in images. They define the energy on the space of boundaries in the image domain and combine different cues: intensity gradient, texture and color. They introduce two polynomial-time directed graph algorithms for finding the global minima of this energy. The first one found the globally optimal solution by using a minimum cut algorithm. The second one uses a minimum ratio cycle algorithm to find a cycle with minimum energy in an embedded graph.

Wang and Siskind [53,54] use a ratio cut to extract salient boundary from a set of noisy boundary fragments detected in real images. It formulates the salient-boundary detection problem into a problem for finding an optimal cycle in an undirected graph. Similar to previous paper, the boundary saliency is defined using the Gestalt laws of closure, proximity and continuity. This paper first constructs an undirected graph with two different sets of edges: solid edges and dashed edges, which means different weight. The most salient boundary is detected by searching for an optimal cycle in this graph with minimum average weight. The most salient boundary

could only be a partial boundary. Instead of multiple objects, right now this paper is only limited to detect single object boundary in an image.

### *2.1.3 Model Based Approach*

The model-based approach is represented by the snake or active contour method [28]. It uses gradient descent to optimize the functions locally on the space of curves in an image. The main advantages of snakes are those of all optimization techniques: the image data, desired contour properties and knowledge-based constraints are integrated into a single extraction process. The disadvantage is that the final extracted contour is highly dependent on the position and shape of the initial contour as consequence of the many local minima in the energy function. This approach also lacks real interactive feedback. The user has to wait until the algorithms converge.

More recently, Elder et al. [10,11] introduce a Bayesian framework for combining prior probabilistic knowledge of an object's appearance with probabilistic models for contour grouping. I would think it also belongs to model based methods, which have some specific object prior knowledge in the form of trained priors. They solve it by using an approximate, constructive search technique, which finds a good solution, but not guaranteed to be optimal. This method has similar limitations as the snake method.

## 2.2. Bayesian Network in Computer Vision

Bayesian networks provide a formalism for reasoning under uncertainty. A Bayesian network is defined by a directed acyclic graph over nodes representing random variables of interest. The arcs signify the existence of direct causal influences between the linked variables. The Bayesian network yields a uniform framework for studying perceptual organization [14].

There has been a lot of work recently that applies graphical models to computer vision and image understanding; e.g, Bayesian networks and Markov Random Fields [3,14,20,26]. Of particular relevance to this work are those methods that use Bayesian networks in computer vision.

### 2.2.1 *Bayesian Networks in Image Interpretation*

Bayesian networks provide a natural framework for computing an interpretation of an image from a set of hypothesized objects. Unlike systems that find interpretations by simply counting matched features, Bayesian networks allow one to exploit the information present in object interactions, both visual and physical. Finding the best interpretation of modeled scene is then equivalent to finding the MPE of the network.

Buxon and Gong [3] describe an application of Bayesian networks to visual surveillance. The task involves the tracking of purposively moving objects and of the dynamic relationships between these objects, such as “overtaking”, “following” in the context of traffic monitoring. A Bayesian network is used to infer object

characteristics from the evidence provided by optical flow measurements. Prior knowledge is used to initialize the network. These objects are tracked by find the Most Probable Explanation (MPE) for the measurements at every frame, and updating the Bayesian network with the evidence provided by these measurements.

Kumar and Desai [29] formulate the image interpretation as the maximum a posteriori (MAP) estimate of a properly defined probability distribution function. They shows that a Bayesian network can be used to represent this probability distribution function as well as the domain knowledge needed for interpretation. The Bayesian network may be relaxed to obtain the set of optimum interpretations.

Kalitzin etc [27] propose a Bayesian grouping approach for recognition and segmentation of large-scale structures representing objects in images. It is based on the detection of local image properties, extraction of simple geometrical primitives, and grouping of these primitives according to probability rules and prior models. It selects a list of subsets of the local primitives and finds the optimum set of model priors that maximizes the likelihood of the model samples representing the selected subsets. In contrast with global recognition methods that classify the whole image, this approach aims at solving the recognition task together with the segmentation task.

Westling and Davis [55] use Bayesian network to represent and calculate the configuration of hypotheses that best interprets the images. The network represents both visual effects, such as the creation and occlusion of image features, and physical constraints such as object interference. Once the network is generated, it can be evaluated to find the set of hypotheses that best account for the image features.

Westling and Davis [56] later cast the image interpretation as the problem of finding most probable explanation (MPE) in a Bayesian network that models both visual and physical object interactions. The network was used in 3-D object understanding to related model components to predict appearances and to control the interpretation process.

### *2.2.2 Bayesian Network in Object Recognition and Segmentation*

Object recognition and segmentation are the perceptual organization problems that integrate low-level image features into high-level object boundary detection. Those problems are well suited problems for the use of Bayesian networks.

Geman and Jedynak [19] track roads in satellite imagery using a decision tree to estimate the Bayesian maximum a posteriori (MAP). They assume independence and a uniform prior. They express road tracking as the maximization of a product of local likelihood ratios.

Coughlan and Yuille [6] also address the problem of tracking roads and develop a tree search that they prove has an expected convergence that is linear in the size of the road (the depth of the tree), though the worst-case performance is exponential.

Tu and Zhu [50,51] have developed a general purpose algorithm for Bayesian inference known as Data-Driven Markov Chain Monte Carlo (DDMCMC). This algorithm has been very successful at segmenting images when evaluated on datasets with specified ground truth. It works by using low level cues to propose high-level



models (scene descriptions) which are validated, or rejected, by generative models. It therefore combines bottom-up and top-down processing in a way that is suggestive of the feedforward and feedback pathways in human brain. The algorithm has been successfully extended to combine segmentation with the detection and recognition of faces and text.

### **2.3. Intelligent Scissors**

Intelligent Scissors is a general purpose, interactive objects selection tool [34,35,36,37,38]. It allows a user to choose a minimum cost contour segment corresponding to a portion of desired object boundary. Intelligent Scissors achieves this goal by an efficient, linear time implementation of Dijkstra's search algorithm on the image graph. As the mouse position comes in proximity to an object edge, the optimal path from the pointer position to the seed points is displayed. It shows like a live-wire boundary snaps to and wraps around the object of interest. The user can select an optimal contour segment. Compared to manual tracing, object selection or segmentation using Intelligent Scissors is many times faster and more accurate.

The initial Intelligent Scissors is pixel based [37]. Currently, it is the toboggan-based Intelligent Scissors with a four parameter edge model [38]. The toboggan-based Intelligent Scissors computes the optimal path by over-segmenting the image using tobogganing and then imposing a weighted planar graph on top of the resulting region boundaries. Because the resulting region-based graph is many times smaller than

previous pixel-based graph, the graph search is faster. By fitting a four parameter edge model, it can even provide sub-pixel localization of object boundary.

As mentioned in the introduction, there are some desired properties for a interactive image segmentation techniques, which are simple, fast, intelligent, general and robust. Intelligent Scissors has those properties. It is simple and fast. It computes and displays the optimal path from the pointer position to the seed point in real time. The user can interactively select an optimal path segment corresponding to a portion of the desired object edge. The pixel-based Intelligent Scissors has two intelligent features: on-the-fly training and boundary cooling. On-the-fly training causes the boundary to adhere to the specific type of edge currently being followed, rather than simply follow the strongest edge in the next. Boundary cooling automatically freezes unchanging segments and automates input of additional seed points [37]. Toboggan-based Intelligent Scissors has live wire extension and confidence based cursor snapping, which reduce both the time required in positioning seed points and the number of seed points required to be input by the user [38].

However, Intelligent Scissors has some limitations [35]. There is still plenty of room for improvement. The first is adding closure constraint. For some simple objects, just snap to the boundary and highlight the whole object with a closed contour. The user does not need to input any seed points and move mouse cursor around the object. Second, the user has to put the seed points in a particular order in Intelligent Scissors. This limits the movement of the mouse to some degree. The style of “live wire” is essentially step by step, which allows the user to select the correct boundary segments

in order and finally make a closed contour. Could we do the other way instead? Show the closed contour in the first place. Let the user fix the contour segments that are incorrect or add the missing correct object boundary. In this way, user can move the mouse cursor around and put the seed points and discard in any order. Third, an intelligent solution or system should not only represent comprehensive and diverse information but also should be flexible enough to incorporate different approaches. It should be a framework. By adding a graphical model to Intelligent Scissors, it allows detecting the object boundary in different ways: with or without a prior object model, with or without evidence or observation. Even possibly change a semi-automatic tool to fully-automatic in some situations.

### 3. Object Highlighting

Object Highlighting is what we name the technique that we are working on. The goal of this technique is to reduce the amount of human effort to near minimum for any user-guided segmentation task. It should define some objects simply by snapping the mouse cursor to the object boundary. The closed contour is displayed on the screen. If it corresponds to the desired object boundary, the contour is selected via a button press. Based on the formulation of our Bayesian network, Object Highlighting could automatically highlight multiple objects without any evidence and highlight single object at a time in images. In this thesis, we only briefly discussed the automatic highlighting since lack of efficient inference algorithm for that situation. We mainly present our algorithm for highlighting single object after user provides some observation.

Since this technique mainly aims to add more intelligence to previous Intelligent Scissors solution, we use the Bayesian networks, a framework in artificial intelligence. Natural images have complexity and ambiguity. Bayesian model allows us to handle these uncertainties by providing both a compact factorized representation and multivariate probabilistic inference. The major challenge of using Bayesian network is keeping the same efficiency as Intelligent Scissors while increasing intelligence. We achieve real time segmentation by formulating our problem and using near linear time graph search to compute single loop MPE.

As Figure 3.1 shows, the process of Object Highlighting technique involves several steps. After inputting the image, it firstly pre-segments the images into tobogganing regions. Edges are the boundaries of those regions and nodes are the junctions between regions. Next, it converts the image into a weighted graph. Based on the graph, it computes the confidence value of each edge in the graph. After those preprocessing steps, it transforms the weighted graph into a Bayesian network. Next, it uses an optimal graph search to compute the MPE of the Bayesian network. MPE of our Bayesian network are the detected loops corresponding to object boundaries. Finally the detected object boundaries are displayed on the image.

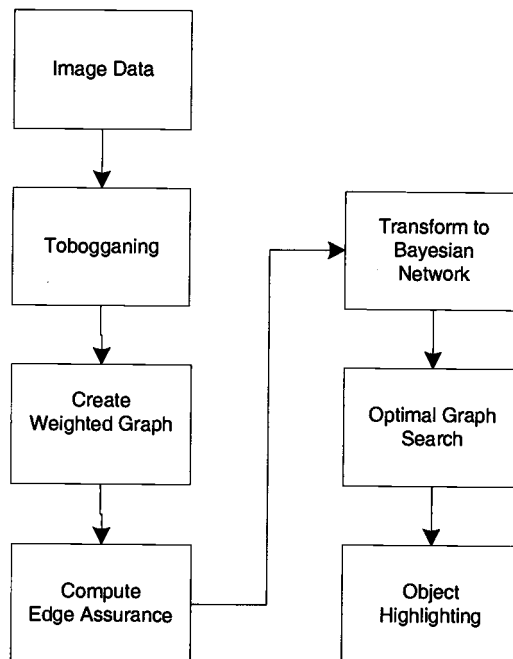


Figure 3.1 An overview of the entire Object Highlighting process.

### 3.1 Image Data Preprocessing

Unlike some previous graph modeling techniques that create graph nodes for every pixel [3,27,45], we build our Bayesian network from the over-segmented region boundaries. As mentioned above, we first partition the image into a collection of small object regions and then impose a weighted planar graph onto the resulting region boundaries [35,38]. Region-based graph provides two advantages over pixel-based graphs: first, it allows the belief network probabilities to incorporate meaningful region based statistics along with the edge-based measures. Region based measure is helpful in making the probability values robust to noise. Second, it reduces the graph size resulting in faster operations.

#### 3.1.1 Tobogganing

Tobogganing technique is effectively identical to computing the watershed of the image's gradient magnitude. Tobogganing over-segments an image into small regions by sliding in the derivative terrain. Figure 3.2 shows the tobogganed regions. The basic idea is that given the gradient magnitude of an image, each pixel determines a slide direction by finding the pixel in a neighborhood with the lowest gradient magnitude. Pixels that "slide" to the same local minimum are grouped together to form small regions. Each region consists of the pixels that "flow" into that region's local minimum [35,38].

Tobogganed regions are similar to the catchment basins produced by applying the popular watershed algorithm [1,2]. However, tobogganing is much more computationally efficient. During tobogganing, each image pixel is processed only once and thus accounting for linear time execution.

### 3.1.2 Weighted Graph Creation

We then impose an undirected graph  $G = (V, E)$  on the resulting boundaries of Tobogganed regions. Vertices  $V = \{v_1, v_2, \dots, v_n\}$  are placed where three or more region boundaries converge, which locate on a pixel corner. Undirected edges  $E = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$  are created along each section of a region boundary connecting two vertices. Since region boundaries follow pixel cracks, every pixel belongs to some region and each vertex has at most degree four. Figure 3.2 illustrates Tobogganing and creation of the weighted graph from an image.

By using different measures applying Gestalt law of perceptual organization, we impose cost value or weights on the edges. There are three measures: gradient magnitude, local region similarity and local edge curvature. Local edge costs are a weighted sum of these three measures. The resulting weighted graph is used as a cost map for object boundary in the image. Thus, graph edges corresponding to object boundaries should exhibit low weights relative to non-object edges. The details of three measurements are summarized in the dissertation [35]. To help understand the work in this thesis, the three measurements are briefly discussed:

*Gradient magnitude:* Gradient magnitude is the local edge strength of the discontinuity in color and brightness. A strong edge will exhibit a high gradient magnitude and therefore should have a low cost.

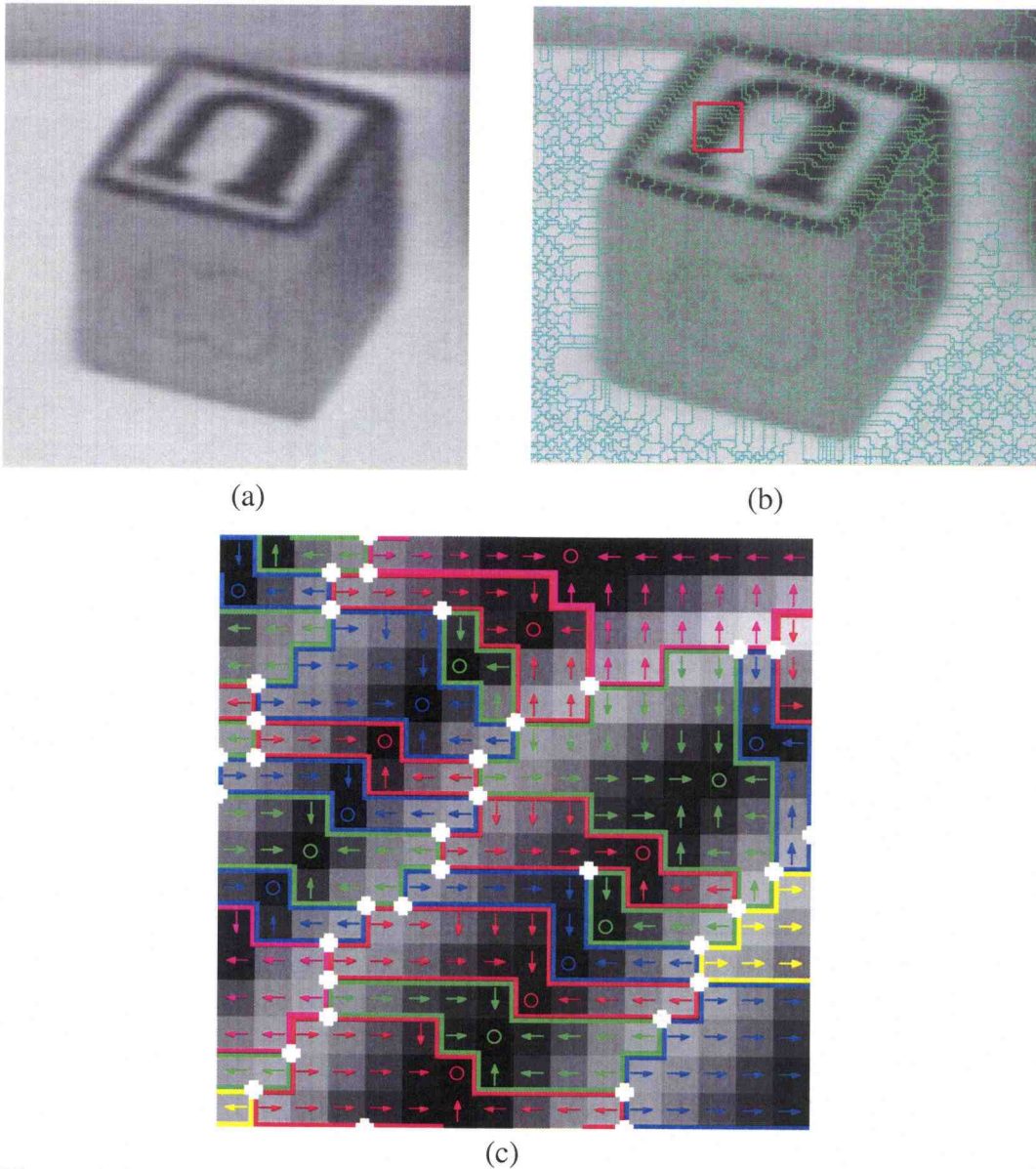


Figure 3.2: Examples of Tobogganing and weighted graph creation. (a) The original 256x256 block image. (b) Tobogganing of the image. (c) Tobogganing regions in a small zoomed area. The colored lines are region boundaries. The small circle in each region is the local minimum. The arrows show the slide direction. (c) also shows how the weighted graph is created. The region boundaries are the edges and the white dots on the junctions are the nodes in the



graph.

*Local region similarity:* This measure indicates statistical similarity between the regions on either side of the edge. The measure currently used is the student's t-test, which measures the probability that two sampled unimodal Gaussian distributions have the same mean.

*Local edge curvature:* The curvature or bending measure adds a smoothness constraint to object contours by associating a relatively high cost for sharp changes in boundary direction. The primary advantage of a bending constraint is to force smooth contours in areas of low contrast or edge strength where noise dominates gradient-based costs.

### 3.1.3 Computing Edge Confidence

Based on previous local edge and region information, we assign to each edge in the weight graph a “confidence” value between 0 and 1.0. An edge's initial confidence is close to 1.0 for edges that are estimated to be on the object boundary and close to 0 for edges that are not.

In previous work, it was called the edge confidence value [35,38]. The edge confidence measure, in turn, relies on another measure called the branch extension cost, which computes the minimum cumulative cost path from among all possible fixed length paths extending out from an edge. For details, please refer to the work of Toboggan based Intelligent Scissors [35,38].

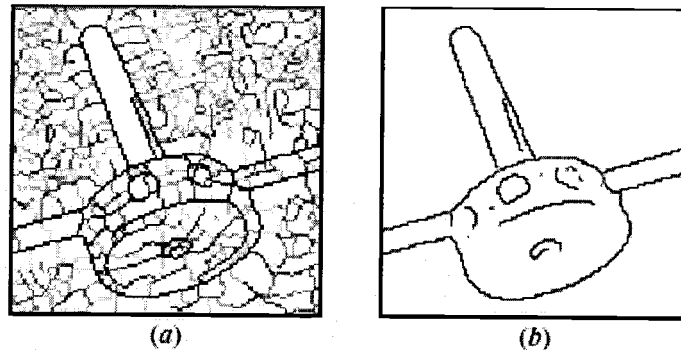


Figure 3.3 shows the edge confidence map for the image in Figure 3.2 (a). (a) Threshold edge confidence at 0.5. All edge confidence larger than 0.5 are shown in black in Figure 3.3 (b).

### 3.2 Bayesian Network Overview

Bayesian networks, also called belief networks, provide formalism for reasoning under uncertainty. A belief network is defined by a directed acyclic graph over nodes representing random variables of interest. The arcs signify the existence of direct causal influences between the linked variables, and the strength of these influences are quantified by conditional probabilities. A belief network relies on the notion of a directed graph. Next are the definitions of Graph and Bayesian network [8].

- **DEFINITION 1: [Graphs]** A directed graph is a pair  $G = (V, E)$ , where  $V = \{X_1, \dots, X_n\}$  is a set of vertices and  $E = \{(X_i, X_j) \mid X_i, X_j \in V, i \neq j\}$  is a set of edges.
- **DEFINITION 2: [Bayesian network]** A Bayesian network is a pair  $(G, P)$ , where  $G = (X, L)$  is a directed acyclic graph over nodes. Let  $X = \{X_1, \dots, X_n\}$  be a set of random variables, each having a finite number of possible states.  $L$

are links or arcs to model direct causal relationship between the variables.  $P = \{P(X_i | Pa_i) \mid i=1, \dots, n\}$  is a set of conditional probability tables (CPT) defined for each variable  $X_i$  and its parents  $Pa_i$  in the  $G$ .

Why using Bayesian network in computer vision? There are three advantages [14,50,51,55,56]. First, Bayesian networks provide formalism for reasoning under uncertainty. A good vision solution should be able to combine large amounts of objectively ambiguous information to yield something that are rarely ambiguous. Second, Bayesian networks provide a joint probability of different factors and good inference algorithms that exploit this structure. Bayesian network could break those complicated computer vision problems down into small problems. Third, Bayesian network emphasizes the role of the generative model, and thus tie naturally to the growing body of work on graphical models.

### 3.3 Our Bayesian Network Model

We describe the creation and notation of our Bayesian network. Same as others, our Bayesian network is a pair of  $(G, P)$ .  $G$  is a directed acyclic graph (DAG) built from the previous undirected weighted graph. The graph is in the form  $G = (X, L)$ .  $X$  is the set of nodes which corresponding to random variables, each having two states: true or false.  $L$  is the set of links that model direct causal relationship between the variables.  $P$  is a set of conditional probability tables (CPT) defined for each variables  $X_i$  and its parents  $Pa_i$  in the  $G$ . Figure 3.4 illustrates how our Bayesian network is created.

The variables  $X_i$  are composed of two types: edge nodes  $X^E$  and vertex nodes  $X^V$ . Both nodes are directly transformed from the weight graph  $G = (V, E)$ .  $X^V$ , created from  $V = \{v_1, \dots, v_n\}$ , is a set of vertices.  $X^E$ , created from  $E = \{e_k \mid e_k = (v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$ , is a set of edges. The links connect the edge nodes and vertex nodes. They are created by the edge nodes  $X_{i,j}^E = \{(X_i^V, X_j^V), i \neq j\}$  connecting its two vertices  $X_i^V, X_j^V$  at both ends. Each  $E_i$  is connected to exactly two vertices. Two links are created on each edge node  $L = \{(X_{i,j}^E, X_i^V), (X_{i,j}^E, X_j^V)\}$ , which connect to the two vertex nodes on both end. The arrow is pointed to the vertex node from the edge node. The intuitive meaning is that edge node has a direct influence on its nodes on both ends. Vertex nodes are created at the junction of neighboring edges.

Besides  $G$ , the other part of Bayesian network is  $P$ , which is a set of CPT, and for each variable  $X_i$ . This set of tables represents the joint probability distribution over the set of random variables  $X$ . Like the set of nodes  $X$ , our set of probability tables  $P = P^E \cup P^V$  is the union of the set of prior probability tables  $P^E = \{P(X_{i,j}^E) \mid X_{i,j}^E \in X^E\}$  and the set of CPT  $P^V = \{P(X_i^V \mid Pa_i) \mid X_i^V \in X^V\}$ .

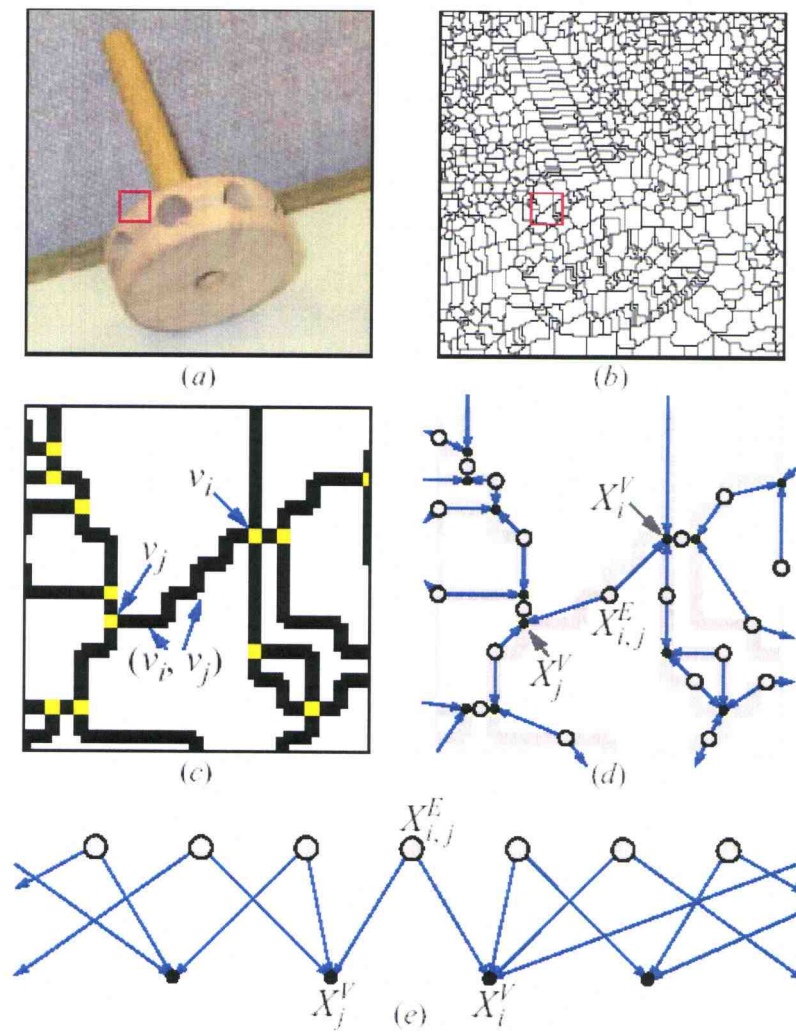


Figure 3.4: Creation of our Bayesian network. (a) Original image. (b) Watershed transform of gradient magnitude. (c) Zoomed section of (b) showing initial graph construction. (d) Belief network constructed from initial graph in (c). (e) Two-layer representation.

Let lower case letters,  $e_k$  and  $v_j$ , denote the values of the corresponding variables  $X_{i,j}^E$  and  $X_i^V$  respectively.  $S_i = (e_o, e_1, \dots e_n)$  denote a particular assignment of edge node states, true or false. The closed boundary of object is a state of the network with a series of edges on the path set to be true and the rest set to be false. There are totally  $2^n$  possible states of the network. The joint probability of our Bayesian network can be written as follows:

$$P(e,v) = \prod_{i=1}^n P(e_i) \prod_{j=0}^m P(v_j = \text{true} \mid pa(v_j)) \quad (3.1)$$

Where  $P(v_j = \text{true} \mid pa(v_j))$  is the conditional probability distribution (CPD) of nodes  $V_j$  given the set of its parents. The parent  $pa(v_j)$  of  $V_j$  is the edge connected to that node in the image weighted graph.  $P(e_i)$  is the prior probability that  $E_i = e_i$  before any edge has been set to true.

### 3.3.1 The Two Layer Topology

The topology of our graph model is a planar directed acyclic graph. The top layer has all the edge nodes (*E-node*), which are the parent nodes holding prior information. The second layer has all the vertex nodes (*V-node*), which are the conditional variables. The two layers are connected by the arcs: which model the relationship of *E-nodes* and *V-nodes*. The relationships are: each *E-node* is connected to exactly two *V-nodes* and each *V-node* is connected to three or four *E-nodes*. From Bayesian point of view, we can also think the top layer is hypothesis layer which

include all the hypothesis variables. The second layer is the finding layer which constrains the relationship of those hypothesis variables.

The topology formulation is focusing on the research goal of this thesis: object boundary detection. The two layer topology is one of the simplest structures. First, this simple structure could simplify the computational complexity of Bayesian network. Second, this structure is also able to capture the relationships of edges segments in an image. An example of simple Bayesian network topology is shown on Figure 3.5.

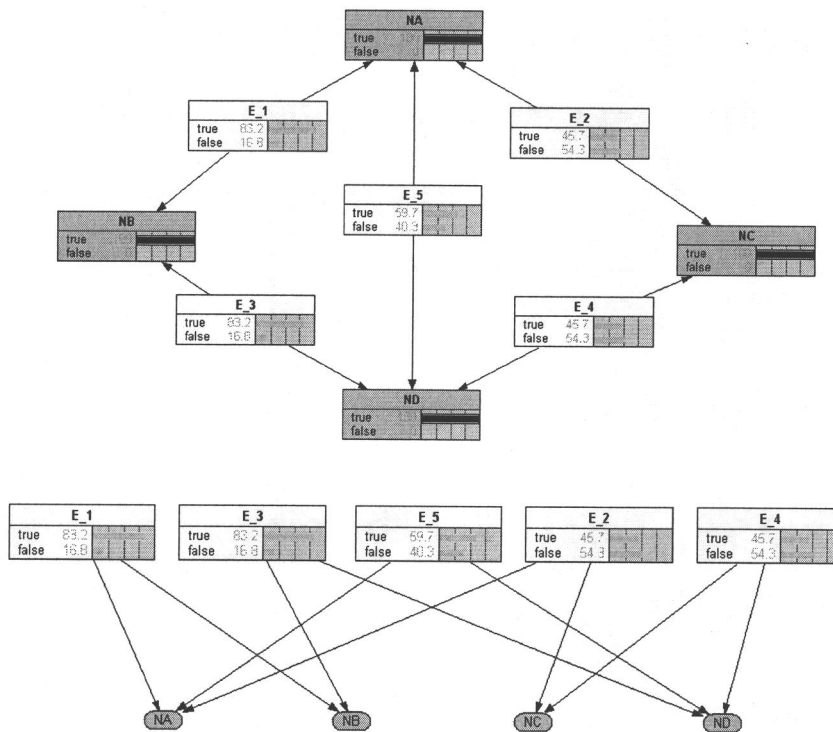


Figure 3.5: Two layer graph topology example in Netica format.

### 3.3.2 Probability Tables

The formulation of probability tables of both nodes and edges is based on the confidence value of each edge. As a general segmentation strategy, there should be no restriction on the size, shape and type of the object. The way we design the probability table is that it should not include application-specific priors, but could add them if desired later.

The goal of our technique is to find complete, closed contours in the image that correspond to object or semi-object boundaries. Bayesian network is the way to decompose the global hypothesis, the closed object contour, to small local hypothesis, the edge segments. As such, our design of *V-node* and *E-node*'s probability tables should satisfy the following criteria:

- *Local edge probability*: estimate the probability of each local edge segment corresponding to object boundaries.
- *Closed contour*: Incorporate low level knowledge of object boundary about how to connect the segments into closed contour. Enforce closed contours.
- *Connecting local probabilities*: Link the decomposed local hypothesis to a global solution. When given evidence for one *E-node*, the observation should be able to propagate to all the other *E-nodes*. This could be done by the neighboring relationships of the edges in the graph.

In the following, the probability table of *E-node* and *V-node* are discussed separately. The *local edge probability* requirement is handled by the probability table



of *E-node*. The *closed contour* and *connecting local probability* are handled by the conditional probability table of *V-nodes*.

### 3.3.2.1 Probability table of E-node

Probability table of E-node is the local prior probability of the Bayesian network based on the edge confidence value. For each edge in the image region graph, we define a Boolean variable  $E_i$ . If edge  $i$  is on the most probable loop, then the value of  $E_i$  is true; otherwise,  $E_i$  is false. The prior in the edge table is formulated as ratio of the paths passing through current edge and not passing through current edge.

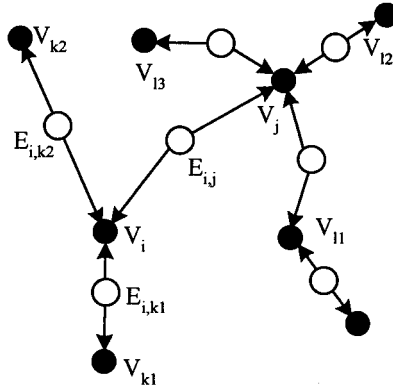
The first idea we have is to define the true probability of an *E-node* as its confidence value normalized by the sum of all edges in the image. The sum of all the edge segments in the weighted image graph has a probability 1.0. This is shown in Equation (3.2).

$$P(X_{i,j}^E = True) = \frac{a(v_i, v_j)}{\sum_{(v_k, v_l) \in E} a(v_k, v_l)} \quad (3.2)$$

This formulation of *E-node* probability turns out to be not very helpful to our research problem. There are two reasons. First, this formulation produces very small probabilities for even the strongest edges and as such, an *E-node* is more likely to be off than on. This means that when a loop is forced to exist in response to some observations, the maximum probability loop will try to turn on as few *E-node* as possible. The results are small loops that typically surround only pre-segmentation regions. Second, Equation (3.2) shows only one edge can be on or only edge which is

locally strong has more probability. For the case of all weak priors on some place of the image that lack contrast, they still could have object boundary. Equation (3.2) does not model this situation.

The low level knowledge of object boundary shows that boundary should be relatively strong in dissimilarity of local region and should be continuous even in the part that lost contrast. Since we have already modeled the local contrast of the *E-node* from its local confidence, we should model the edge by their continuity of the confidence value relative to its neighboring *E-nodes*. An important piece of knowledge for normal object boundaries is that all the E-nodes along the object boundaries should be spatially continuous and should have exactly degrees of two.



	True	False
Edge $X_{i,j}^E$	$P(X_{i,j}^E = True)$ Equation (3.3)	$1 - P(X_{i,j}^E = True)$

Figure 3.6: Table for E-Node.

$$P(X_{i,j}^E = True) = \frac{Sum_{on}(i, j)}{Sum_{on}(i, j) + Sum_{off}(i, j)} a(v_i, v_j) \quad (3.3)$$

$$Sum_{on}(i, j) = \frac{1}{3} \sum_{X_{i,(k \neq j)}^E \in pa_i} \left\{ \sum_{X_{(l \neq i), j}^E \in pa_j} a(v_k, v_l) + a(v_i, v_j) + a(v_j, v_l) \right\} \quad (3.4)$$

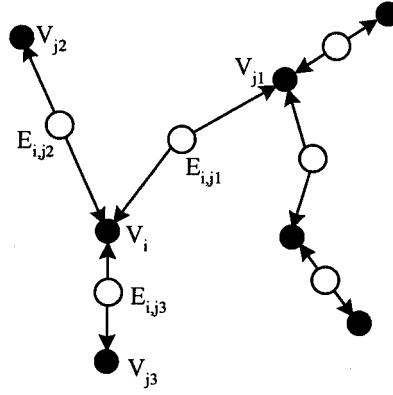
$$Sum_{off}(i, j) = \frac{1}{2} \sum_{X_{i,(k \neq j)}^E \in pa_i} \left\{ \sum_{X_{i,(l < k, l \neq j)}^E \in pa_i} a(v_k, v_l) + a(v_i, v_l) \right\} \\ + \frac{1}{2} \sum_{X_{(k \neq i), j}^E \in pa_j} \left\{ \sum_{X_{(l < k, l \neq i), j}^E \in pa_j} a(v_k, v_j) + a(v_j, v_l) \right\} \quad (3.5)$$

Equation (3.3), (3.4) and (3.5) show how the E-node probability is computed. Equation (3.3) shows the ratio of the weighted total confidence of all the paths that pass through that edge to all paths that not pass through that edge in the neighborhood. Equation (3.4) is the summed confidence of all neighborhood paths that pass through edge( $v_i, v_j$ ). Equation (3.5) is the summed confidence of all neighborhood paths that do *not* pass through edge( $v_i, v_j$ ).

The ratio of weighted sums in Equation (3.3) assigns a high probability to an edge if it is on one or more high confidence paths that pass through that edge's neighborhood. One problem is that it did not differentiate the situation of noise and real object boundary. In some noise situation, it could happen that all the edges have low confidence but one or two have relatively higher confidence. To differentiate these two situations, we choose to multiply the ratio by the edge's initial confidence value.

### 3.3.2.2 Probability table of V-node

From the Gestalt law of perceptual organization, we know that closure and local feature continuity are often considered important features to ensure object boundary. Both these constraints can be enforced by the CPT of the *V-node*. To demonstrate how we designed the CPT of V-node, Figure 3.7 shows the CPT for a degree three *V-node*.



$E_{ij1}$	$E_{ij2}$	$E_{ij3}$	$P(X_i^V = True)$
<i>False</i>	<i>False</i>	<i>False</i>	0.5
<i>False</i>	<i>False</i>	<i>True</i>	0.0
<i>False</i>	<i>True</i>	<i>False</i>	0.0
<i>False</i>	<i>True</i>	<i>True</i>	$P_i^3(j_2, j_3; j_1) / Sum_i^3(j_1, j_2, j_3)$
<i>True</i>	<i>False</i>	<i>False</i>	0.0
<i>True</i>	<i>False</i>	<i>True</i>	$P_i^3(j_1, j_3; j_2) / Sum_i^3(j_1, j_2, j_3)$
<i>True</i>	<i>True</i>	<i>False</i>	$P_i^3(j_1, j_2; j_3) / Sum_i^3(j_1, j_2, j_3)$
<i>True</i>	<i>True</i>	<i>True</i>	0.0

Figure 3.7: Table for V-node

$$P_i^3(j_1, j_2; j_3) = a(v_i, v_{j_1})a(v_i, v_{j_2})[1 - a(v_i, v_{j_3})] \quad (3.6)$$

$$Sum_i^3(j_1, j_2, j_3) = \sum_{k=1}^3 P_i^3(j_{(k+1) \bmod 3}, j_{(k+2) \bmod 3}; j_k) \quad (3.7)$$

Degree of  $v_i$  is 3. Degree of  $v_{j_1}$  is 4.

Equation (3.6) is the unnormalized probability of the assignment (true, true, false) to  $E_{i,j_1}$ ,  $E_{i,j_2}$  and  $E_{i,j_3}$  respectively. Equation (3.7) is the normalization factor which sum over the probability of all non-zero entries of node  $v_i$ .

The CPT for a degree four V-node is similar except that it has  $2^4=16$  rows, 7 non-zero entries (including the false case). The Equation (3.6) and (3.7) can be easily change to four degrees  $P_i^3(j_1, j_2, j_3; j_4)$  and  $Sum_i^4(j_1, j_2, j_3, j_4)$

### 3.3.2.2.1 Connecting local probability

In the image weighted graph, vertices are the junction of edges. *V-node* connects the local prior probability of the *E-node*, which means the influence among *E-nodes* happens via *V-nodes*. After transforming the image weighted graph into Bayesian network, the V-nodes are the finding nodes that are known and constrain the probability relationship and the way of propagation of E-node's observation.

The probability of all the *V-nodes* is set to *true*, which means they are the finding nodes. This is very important because it create the influence between edges. If all the V-nodes are set to *false*, this is called “D-separation” in Bayesian network terms. “D-separation” means that no information can be transmitted between the two nodes X and Y given a set of nodes Z while all paths between X and Y are blocked by Z. All the *E-nodes* are conditional independent and not influence each other any more.

The change of local probability of *E-node* would not propagate to other *E-nodes*. If all the V-nodes are set to *true*, the *E-nodes* are not D-separation anymore. The local neighboring *E-nodes* have direct influence to each other. For the change of probability of a single *E-node*, it is possible to affect any other *E-nodes* in the Bayesian network. This is done by propagating the evidence along a specific path through neighboring V-nodes. This assumption is known as *causal independence* [23].

### 3.3.2.2.2 Closed contour

A contour in the image is represented as either a path in the Bayesian network graph  $G$  or a sequence of married E-nodes where two E-nodes are married if they are both parents to the same V-node. The object boundary is a closed contour which is represented as a cycle in graph  $G$  or a sequence of married E-nodes such that  $i_m = i_1$  in Equation (3.8). The first and the last edge in the sequence are neighbors. Each edge segment in a closed contour has exactly degree two.

$$(X_{i_1, i_2}^E, X_{i_2, i_3}^E, X_{i_3, i_4}^E, \dots, X_{i_{m-2}, i_{m-1}}^E, X_{i_{m-1}, i_m}^E) \quad (3.8)$$

Due to the relational nature of our image weighted graph, if an edge  $i$  is on the most probable loop, then one and only one edge from both sides of its neighbors is on the most probable loop. In the CPT entries, we can model these relationships by allowing the cases of two edges is true could happen and not allow other cases happen, like one, three or four edges are true. This means the true probability of the entries are non-zero if and only if there are only two neighboring E-nodes are true. We also model the case that all the edges are false to occur. Since there is no other information for us to tell how likely this case could happen, the probability is set to be 0.5 which

means it does not favor either situation. Above setups on CPT of V-nodes could enforce simple, non self intersection and non overlap closed contours.

### 3.4 Bayesian Inference

Inference refers to the process of computing the answer to a query in the Bayesian network. There are two main types of Bayesian Inference tasks: belief updating and belief revision [39,43]. The objective of belief updating is to calculate  $P(H|E)$ , the posterior probabilities of query nodes  $H$ , given some observed values of evidence nodes  $E$ . The task of *belief revision* is about finding the most probable instantiation of all or some of the hypothesis variables, given the observed evidence [21,7]. There are two types of belief revision: most probable explanation (MPE) and maximum a posterior hypothesis (MAP). MPE is about finding maximum probability assignment to all hypothesis variables given the evidence. MAP is more general query and requires finding a maximum probability assignment to a subset of hypothesis variables given the evidence. Although MAP query is more general, MPE is a special case because it is computationally simpler and thus should be applied when appropriate [8].

Researchers have developed various kinds of exact and approximate Bayesian network inference algorithms. Some of them are particularly designed for certain applications. Next, we briefly summarize three popular Bayesian Inference algorithms. For *polytree* like structure, there is an efficient message propagation inference algorithm. It is exact and polynomial complexity in the number of nodes [39,43]. The

most popular exact Bayesian network inference algorithm is *Clique-tree* message passing algorithm [30,25]. It is also called “Junction tree” algorithm. It works efficiently for sparse networks. Its complexity is exponential in the size of the largest clique of the transformed undirected graph. Symbolic probabilistic inference (SPI) views probabilistic inference as a combinatorial optimization problem, the optimal factoring problem. The performance of different Bayesian inference algorithms depend on the network characteristics: size, connectivity, skewness of the conditional probability table [21].

Inference algorithms for arbitrary Bayesian networks are impractical for large, complex belief networks. However, Inference algorithms for specialized classes of Bayesian networks have been shown to be more efficient. Huang and Henrion present an efficient search-based algorithm for approximate inference on arbitrary, noisy-OR belief networks [24].

### 3.4.1 MPE in Our Bayesian Network

Once the Bayesian network is specified, boundary detection can be formulated as finding the MPE that is the most likely assignment to all  $E_i$  nodes in our Bayesian network given some observations  $\{E_{j1}, E_{j2}, \dots, E_{jn} = \text{true}\}$ . In the MPE, all the edges on the object boundary should be set to true.

$$E^* = \operatorname{argmax} P(e, v) = \operatorname{argmax}_e \prod_{i=1}^n P(e_i) \prod_{j=0}^m P(v_j = \text{true} \mid pa(v_j)) \quad (3.9)$$



The problem of how to determine exact conditional probabilities for the network is shown to be unimportant, since the goal is to find the most probable configuration of edges in the graph, not to calculate absolute probabilities. The most probable configuration of edges is a closed contour in the image or loop in the graph.

Determining the general case MPE is known to be NP-hard [5]. Recently, it has been proved that even finding the MPE of a two-level belief network in general is NP-hard [49]. After transforming weighted image graph into our Bayesian network, the size is very large. For example, a general real 256 by 256 image has more than 36,000 nodes, including both the *E-nodes* and *V-nodes*. In the following, we show two methods for finding the MPE in our Bayesian network. The first is computing the exact MPE without evidence using available inference algorithm and software. The second method is an efficient MPE algorithm we developed for this problem, which specifically designed for finding single loop with evidence in our Bayesian network.

### 3.4.2 MPE without Evidence

After formulating our research problem into Bayesian network inference, we first want to compute the exact MPE without evidence. To solve this problem, we use the well known Bayesian Inference algorithm and available free/commercial library. Since our network is very sparse, *clique tree* message passing algorithm, the fastest known exact Bayesian Inference algorithm, should be the best choice [21].

We use *Netica* for computing the exact MPE, which is the world's most widely used Bayesian network commercial software. It uses the *clique tree* message passing

algorithm for exact probabilistic inference in a compiled Bayesian network [41]. “Compile” means transforming the Bayesian network into a large set of data structure called “junction tree”. Bayesian Inference is done by the message passing algorithm operating on the underlying junction tree structures. The size of Bayesian network that Netica can handle depends on the amount of memory required for compiling the whole network into junction tree structures.

There are advantages and disadvantages of using Netica. The advantages are that it is robust and standard. It is not an easy task to implement the *junction tree* message passing algorithm from scratch by ourselves. We also need to make sure the *junction tree* message passing algorithm could practically handle our problem. The disadvantage is that it is a general algorithm and not specifically designed for our problem. When our network is larger than a certain size, Netica can not compile it into a junction tree structure due the huge memory requirement.

To find out the computation limitation of Netica on our problem, we tested different sizes of our Bayesian network. The maximum number of nodes that Netica can handle is about 1300 nodes totally. When it is above that range, Netica will report an error indicating shortage of memory during complication. Figure 3.9 shows an example of exact MPE computed by Netica. Without any evidence, there are three loops automatically highlight three object boundaries respectively. Figure 3.8 shows an example of automatic highlighting, computing MPE without evidence, with the size of 1044 nodes in total.

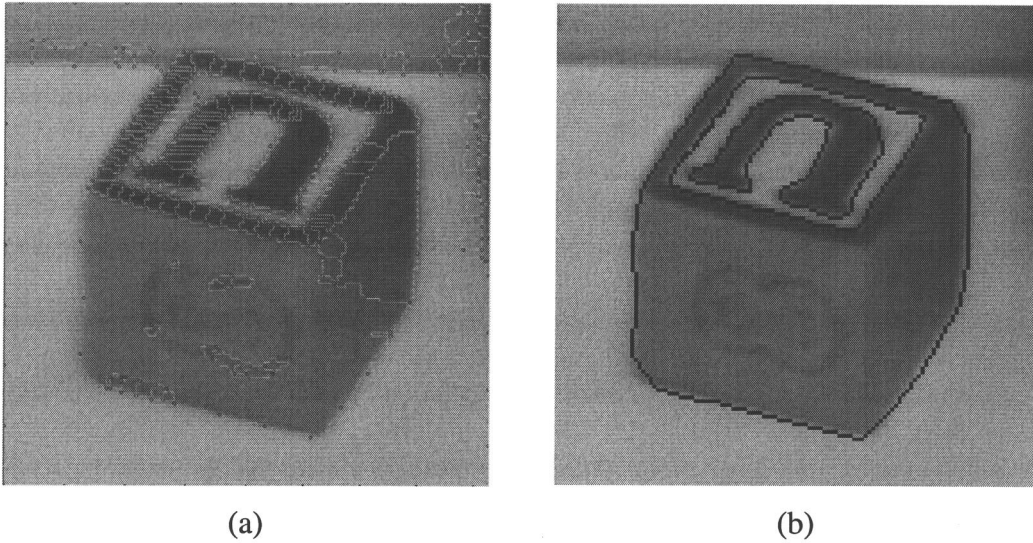


Figure 3.8: An example of MPE without evidence on a 128 by 128 image. (a) The threshold image graph. The number of V-node is 422. The number of E-node is 622. The gradient magnitude of Tobogganing is threshold at 1/50 of the maximum possible gradient magnitude. The right figure (b) shows the three loops detected by computing the exact MPE, which correspond to the object boundaries.

### 3.4.3 Single Loop MPE with Evidence

#### 3.4.3.1 Simply the problem by finding single loop

Although we can compute the MPE with existing Bayesian Inference algorithms and libraries, it is not practical. The algorithm does not scale up and can only handle limited size. As we mentioned earlier, the computation of object boundary need to be fast and hopefully work real time in an interactive segmentation environment. We need to develop our own inference algorithm, which is efficient, scales up and specially designed for our Bayesian network. The customized inference algorithm should be able to compute MPE of our Bayesian network with thousands of

node in fraction of second in a general case, like 20,000 nodes. It should also be able to scale up in the situation of large size images.

Based on the specialized characters of our Bayesian network, we think there may be a polynomial time algorithm that computing exact MPE for multiple loops without evidence. It is due to two reasons. First, our network is very sparse, all the *E-nodes* have degree of two and *V-nodes* have degree of three or four. Second, our network has special designed CPT for *V-node*, which enforces simple, closed contours. Though we have not got an efficient algorithm now, we put this on our list of future works.

Other than working on efficient solution for this complex problem, we tried another approach, which is reducing the computational complexity based on our research goal. The major goal of this research is to find the object boundary in an image for the users before or after they provide some observation. In an interactive environment, we find that user usually only select one object at a time. Based on this assumption, we simplify our problem to finding single loop in an image given some evidence, though it is possible that there are multiple objects forming multiple loops in an image.

### 3.4.3.2 Non-negative graph weights

We create a weighted, planar graph by simply adding edge and vertex weights to our initial planar graph described in Section 3.1 from the corresponding probability tables in the belief network. Since an E-node is assumed to be *off* prior to any

evidence, an edge weight  $w^e(v_i, v_j)$  corresponds to flipping an E-node from *off* to *on* and is given by

$$w^e(v_i, v_j) = -\ln \left[ \frac{P_{on}^E(i, j)}{P_{off}^E(i, j)} \right] = \ln P_{off}^E(i, j) - \ln P_{on}^E(i, j) \quad (3.10)$$

Likewise, a vertex weight is given by

$$w^v(v_j; v_i, v_k) = -\ln \left[ \frac{P_{on}^V(i, j, k)}{P_{off}^V(j)} \right] = \ln P_{off}^V(j) - \ln P_{on}^V(i, j, k) \quad (3.11)$$

and corresponds to switching the node from having no parents on to having the parents  $X_{i,j}^E$  and  $X_{j,k}^E$  on.

If  $P_{on} > 0.5$  in either Equation 3.10 or 3.11, then the resulting weight will be negative. Search algorithms for undirected graphs with negative weights are more computational than those with non-negative weights [1,37]. In the Intelligent Scissors framework, Dijkstra's shortest path algorithm could be implemented to compute an optimal solution in near linear time  $O(V+E)$  for a graph  $(V, E)$  [36,38]. Dijkstra's algorithm assume that all edge weights in the input graph are nonnegative. Bellman-Ford algorithm allows negative weights in the input graph. It can produce a correct answer as long as no negative weight cycles are reachable from the source. The time complexity is  $O(VE)$ . Due to the large number of V-nodes and E-nodes, Bellman-Ford algorithm is not fast enough for the interactive image segmentation in general.

### 3.4.3.3 Clipping table

For simplifying computation of MPE by allowing for efficient graph-based search techniques, we clip the probability table. Clipping CPT table means suppress the true prior probability value of E-node. When the value is greater than 0.5, we clip the value to 0.5. This provides two advantages over the previous non-clipping CPT table. Other than allowing using efficient shortest path search to compute the MPE, it also provides length invariance for the path expansion or observation propagation.

Clipping probability tables of both the *V-node* and *E-node* in our Bayesian network make the cost of edges during the graph search non-negative. All the *False* probability is 0.5 in our CPT table. The maximum clipped *True* probability is 0.5. The cost of flip an edge to on is always going to be positive or zero. This enable us to do the Dijkstra's graph search.

The clipped tables suppress all the edges that are false as the initial MPE assignment, which means there is no user's evidence. We can think all the *E-nodes* are off before user's interaction. When the user provides evidence, which turn one edge on by setting it to *True*, this observation is going to propagate to the whole network.

The clipped probability tables enable length invariance during path expansion to form closed contours. Length invariance means it should neither penalize nor reward an object contour based on its length. Due to clipping the probability table, some edges have equal *true* probability and *false* probability, both at 0.5. The cost of adding this edge to path is zero. This creates long segments of zero cost, which are those continuous high confidence edge sequences. It overcomes the tendency towards

small loops if all the edges have positive cost. It does not favor longer path, because add more high confidence edge sequences does not decrease total cost as the negative weight situation.

### 3.5 Graph Search and Loop Detection

We use a variation of Dijkstra's graph search [37] on the non-negative weight graph to find a single loop corresponding to object boundary. Given a weighted graph and an evidence edge, the edge starts to expand to both sides and a minimum-path spanning tree (MPST) is computed, being rooted at one of the vertex of that edge. Two major branches are formed on both side of the edge and all paths emanate from one of the two endpoints of the evidence edge.

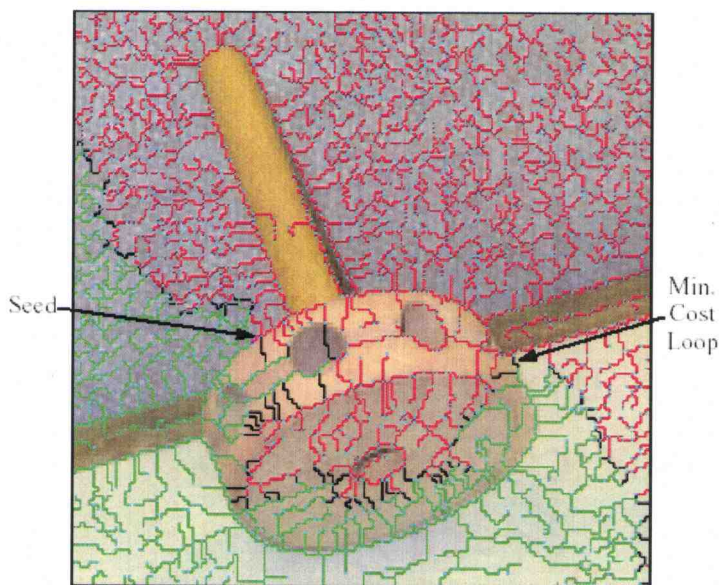


Figure 3.9: Loop detection using a spanning tree. The minimum-path spanning tree of a 128 by 128 image created using Dijkstra's graph search with a seed edge (shown in purple). The two major branches of the seed are shown in red and green while the edges that are excluded from the spanning tree that connect

major branches, thus forming loops that include the seed, are shown in black. (The excluded edge forming the minimum cost loop is indicated).

The spanning tree expands to every node but not every edge. The reason is that all the V-nodes are set to true in the Bayesian network. Excluded edges are the edges that are not included in the spanning tree. An excluded edge connects two nodes from different major branches and forms a loop. Generally, a loop includes edges on branches from the two major branches of the spanning tree, the exclude edge and evidence edge provided by the user. The cost of the loop is the multiplication of clipped prior probability of all the E-nodes and relative probability table entries of the V-node along the loop.

Maximization of multiplication can be transformed into minimization the sum of the negative logarithm of Equation (3.8). Computing the MPE is equivalent to finding the assignment that minimizes Equation (3.12).

$$\begin{aligned}
 -\ln P(e, v) &= \arg \min_e \left\{ -\ln \left[ \prod_{i=1}^n P(e_i) \prod_{j=0}^m P(v_j = \text{true} \mid pa(v_j)) \right] \right\} \\
 &= \arg \min \left\{ -\sum_{i=1}^n \ln P(e_i) - \sum_{j=0}^m \ln P(v_j = \text{true} \mid pa(v_j)) \right\} \quad (3.12)
 \end{aligned}$$

After the evidence is given, the process of MPST propagation is: the observation will propagate to all the V-nodes which are set to be true. When we find an excluded edge that connects two branches, we compute the loop cost and keep track of the minimum cost loop. Closed contour formed by two branches of the search tree. We can propagate to all the nodes of an image, but not all the edges. The minimum



cost loop of different scale is computed during the expansion, which forms small loops to large ones. There is linear number of loops. The best loop is the one with minimum cost among those “minimum cost” loops of different scales.

Following is the Psuedo code for the algorithm of single loop MPE with evidence:

**Algorithm 1:** MPE loop detection.

**Input:**

$P_{seed} = (v_{s_1}, \dots, v_{s_m})$  {Seed path.}

**Data Structures:**

**A** {Active list: sorted by path cost (initially empty).}  
**nghbr(v)** {Set of neighbors of (vertices adjacent to) vertex  $v$ .}  
**done(v)** {Boolean function: *true* if  $v$  has been expanded.}  
**g(v)** {Total path cost function from  $v_s$  to  $v$ .}  
**branch(v)** {Major branch that  $v$  is on (initialized to 0).}  
 **$v_{min}(v)$**  {Neighboring vertex on min cost path from  $v$ .}

**Output:**

$e_{MPE}$  {Edge specifying most probable loop.}  
 $g_{min}$  {Minimum cost of loop formed by  $e_{MPE}$ .}

**Algorithm:**

```

1   $g(P_{seed}) \leftarrow 0$ ;  $g_{min} \leftarrow \infty$       {Init. cost for seed path vertices and min. loop.}
2   $branch(v_{s_1}) \leftarrow 1$ ;                  {Set major branch of seed path end vertices }
3   $branch(v_{s_m}) \leftarrow 2$ ;                  { to different values.}
4   $A \leftarrow v_{s_1}, v_{s_m}$                     {Place seed path end vertices on active list.}
5  while  $A \neq \emptyset$                         {While still nodes to expand:}
6     $v \leftarrow \min(A)$ ;                      {Remove min. cost node  $v$  from active list.}
7     $done(v) \leftarrow \text{TRUE}$ ;                 {Mark  $v$  as expanded (i.e., processed).}
8    for each  $v_i \in \text{nghbr}(v)$               {For each neighboring vertex  $v_i$  of  $v$ : }
9      if not  $done(v_i)$  then                 {If neighbor not yet expanded;}
10          $g' \leftarrow g(v) + w^v(v; v_{min}(v), v_i)$  {Compute cost to neighbor.}
11          $+ w^0(v, v_i)$ ;
12         if  $v_i \in A$  and  $g' < g(v_i)$  then      {Remove higher cost }
13            $v_i \leftarrow A$ ;                     { neighbor from list.}
14         if  $v_i \notin A$  then                     {If neighbor not on list, }
15            $g(v_i) \leftarrow g'$ ;  $opt(v_i) \leftarrow v$ ; { set total cost, set opt. path, }
16            $branch(v_i) \leftarrow branch(v)$         { set major branch, and }
17            $A \leftarrow v_i$ ;                       { place on (or return to) list.}
18         else if  $branch(v_i) \neq branch(v)$  then { $v_i$  expanded: check for loop}
19            $g' \leftarrow g(v) + w^v(v; v_{min}(v), v_i)$  {Compute cost of loop.}
20            $+ g(v_i) + w^v(v_i; v, v_{min}(v_i)) + w^0(v, v_i)$ ;
21           if  $g' < g_{min}$  then                     {If new loop cost is lower;}
22              $e_{MPE} \leftarrow (v, v_i)$ ;  $g_{min} \leftarrow g'$ ; { Set new lower cost loop.}

```

Figure 3.10: The algorithm of single loop MPE with evidence.

During the graph search, the path through a vertex is determined only when that node is removed from the active list and expanded (i.e., the total cost to its neighbors is computed). As such, the total path cost from a vertex  $v$  to a neighboring vertex  $v_i$  is equal to the total cost to, but not including,  $v$ , plus the vertex weight for  $v$ , plus the cost of the edge from  $v$  to  $v_i$  (see line 10 above). If during expansion, the neighboring node  $v_i$  has already been expanded and is on a different major branch (line 17), then the edge  $(v, v_i)$  forms a loop that includes the seed path. The total cost of that loop is equal to sum of the total costs to both vertices,  $g(v)$  and  $g(v_i)$ , plus the vertex costs for both  $v$  and  $v_i$ , plus the edge cost  $we(v, v_i)$  (line 18). If the loop cost is less than the current minimum loop cost (line 19) then the minimum cost loop is updated (line 20).

The computational complexity of our graph search is near linear in the number of graph edges. This attributes to an efficient hash table implementation of the active list that allows near constant time insertion and removal of the vertices (including removal of the minimum cost vertex). Figure 3.11 shows a plot created from several images of the average time to compute the MPST relative to the number of graph edges. These times were measured on a 550 MHz Sun workstation. Note that even for large graphs with approximately 100,000 edges the total computation time is about 1/3 of a sec. For comparison, the image in Figure 3.9 is 128 by 128 pixels, has 4,436 edges, and requires approximately 0.008 seconds to compute the MPST.

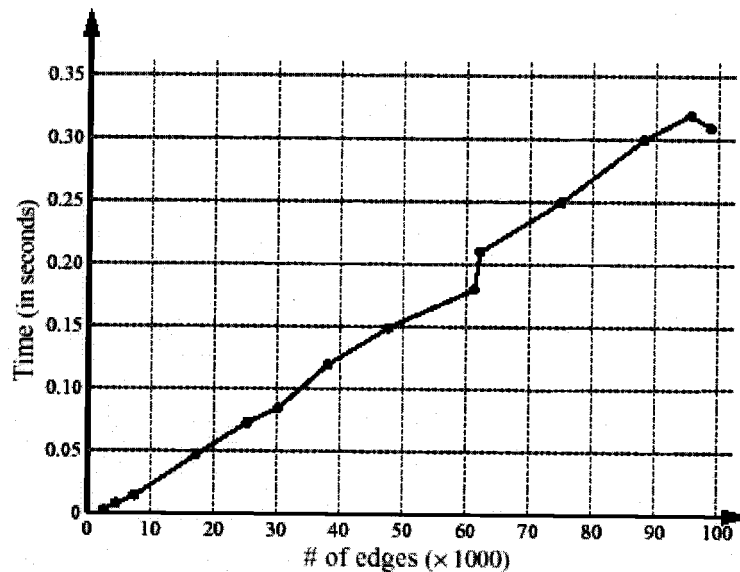


Figure 3.11: Plot showing time to compute the minimum-path spanning tree. The MPE loop of the belief network as a function the number of graph edges or E-nodes in the Bayesian network.

As we discussed earlier, it is important to notice that general Bayesian network Inference task is NP-hard. Our algorithm is capable of computing the approximate MPE in near linear time. This is because the way we simplify our problem as shown in previous section. We assume there is single loop given evidence, clipping the probability tables of E-nodes and V-nodes.

### 3.6 Object Highlighting Interface

Object highlighting is the interface that allows the user to interact with the Bayesian network in real time to quickly specify object boundaries. It allows the user to interact with belief network by providing it with observations and receiving real-time feedback in the form of an updated MPE that displays the most probable loop

based on each new observation. Our Bayesian network and graph search for computing the single loop MPE are all implemented in the Intelligent Scissors framework [35,36,38]. As the user moves mouse around within the image, various objects or subobjects are “highlighted” when cursor snaps to a point on their boundary. We can divide Object Highlighting into several steps:

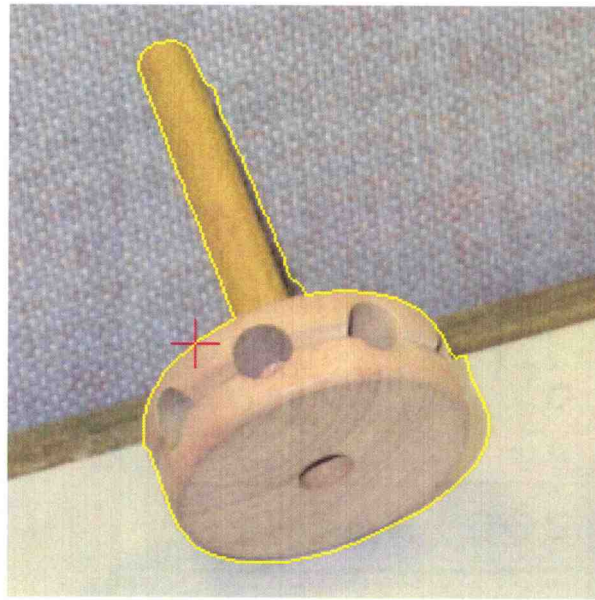


Figure 3.12: Object Highlighting on the object of figures 3.4 and 3.10.

The first step is cursor snapping. When the user moves the mouse, the system automatically snaps the mouse cursor to the edge with the highest confidence and on the boundary of the object that user wants to highlight. The snapped edge or the seed edge, a partial boundary, provides an evidence for the Bayesian network, which means the variable corresponds to that edge will be set true.

The second step is creating the minimum-path spanning tree (MPST) for computing the single loop MPE with evidence in the image. The snapped edge, a

partial boundary, obtained from previous step starts to expand using our graph search. Computation of the MPST is fast enough (even on a 550 MHz workstation) to allow for interactive recomputation of the most probable loop as the mouse moves. Each new cursor position specifies a new evidence edge in the graph causing recomputation of the MPST and display of the resulting MPE loop.

Finally, the detected loop is displayed on the screen. Often, the displayed loop defines the boundary of an object or subobject. If it corresponds to the desired object boundary, the contour is selected by the user to press the mouse button to put an anchored seed point.

## 4. Results and Discussion

This section demonstrates the robustness and generality of our Object Highlighting technique on wide varieties of images including real world color images with complex object boundaries, foregrounds, and backgrounds. All the results are obtained using the single loop MPE method in section 3.4.3 with clipping tables of both *E-nodes* and *V-nodes*.

Figures 4.1 to 4.5 demonstrate how our technique highlights the objects in different images by single observation. In terms of speed, all the objects can be highlighted in less than a fraction of second. As the user moves the mouse cursor close to the target object boundary, the mouse cursor snaps to the boundary and highlights the whole object instantly.

Single observation does not always work. For more complex object, foreground or background, users need to provide multiple observations. Figures 4.6 to 4.10 demonstrate Object Highlighting's performance on images with varying degree of complexity with multiple observations.

In all the figures in this section, the contours, crosshair and small circles are overlaid on top of each image to indicate the object's boundaries and the observations provided by the users. The blue closed contour corresponds to the final detected object boundary. The yellow closed contour is the current highlighted object boundary. The red crosshair is the current mouse position, which provides an observation on the edge

of that position. The small blue circles are the seed points to represent the observations anchored on the object boundary.

## 4.1 Single Observation

All images in this section demonstrate defining an object boundary using single observation. The images are chosen for demonstrating different kinds of difficult situations for normal object segmentation methods.

Let's consider Figure 4.1 to illustrate the process for highlighting the pocket knife with single observation. The process is the same as the one described in the section 3.6. Figure 4.6a shows the cursor snapped to the object boundary to provide the first observation. If the boundary is correct, the closed contour can be selected via a button press, which put an anchor seed point on the boundary of object. Figure 4.6b shows the selected closed contour in blue and the seed point as a red circle.

Figure 4.2 shows two synthetic test images with different shapes, which are created to test how Object Highlighting performs in the presence of different shapes, edge blurring, and white Gaussian noise. Figure 4.2a is a polygonal shape with sharp corners and Figure 4.2b is a curved shape with varying degrees of curvature. Both the test images have white noise with 32 gray levels standard deviation and are blurred using a Gaussian filter with 2.33 pixel standard deviation. Figure 4.2b would be a challenging case for the most recent salient measure techniques [40] for object segmentation, due to the small edge segments and lack of smoothness and convexity in the part with varying degrees of curvatures.

Figure 4.3 shows three object boundaries on the block image and one object boundary on the hat image. Compared to Figure 3.8 that highlights three object boundaries automatically, we can see the result boundaries are the same in Figure 4.3a, 4.3b and 4.3c. In Figure 4.3a, the “U” on top of the block demonstrates the length invariance property of our boundary detection algorithm. If the algorithm favors longer loops, the loop of the “U” will connect with the outside rectangular boundary shown in Figure 4.3b. If it favors a short loop, it will just generate some small local loops around the tobogganing regions. Figure 4.3d shows the utility of contour closure constraint. There are other loops around the hat, but those loops are open, like the shadow areas.

Figure 4.4 shows the boundaries of stones on pavement. This figure illustrates how our Object Highlighting produces object boundaries in highly textured background, which are difficult for normal edge following techniques. Notice, the best loop is still the object boundary though the shadow could form alternative loops. For the same image, the most recent salient measure could detect the rough boundary of the three stones in about 10 seconds [40]. Our Object Highlighting only takes a fraction of second to correctly highlight each one.

Figure 4.5 is the color still life image. It demonstrates Object Highlighting’s generality and application to real world color scenes. The yellow rose in Figure 4.5b is selected with a single observation and the red rose in Figure 4.5c is highlighted with two observations. Notice, both roses have inner complexity and are on highly textured backgrounds. The apple in Figure 4.5e is also on a highly textured background.



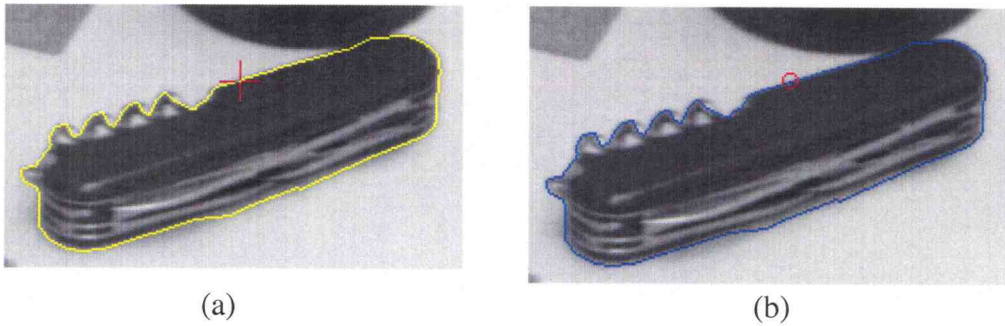


Figure 4.1 Grayscale pocket knife boundary defined with single observations. (a) Provide the first the observation. The whole boundary is selected. (b) If the displayed boundary is correct, press mouse button and the boundary was “set” indicating by the blue contour. The red circle marks the location of the seed edge.

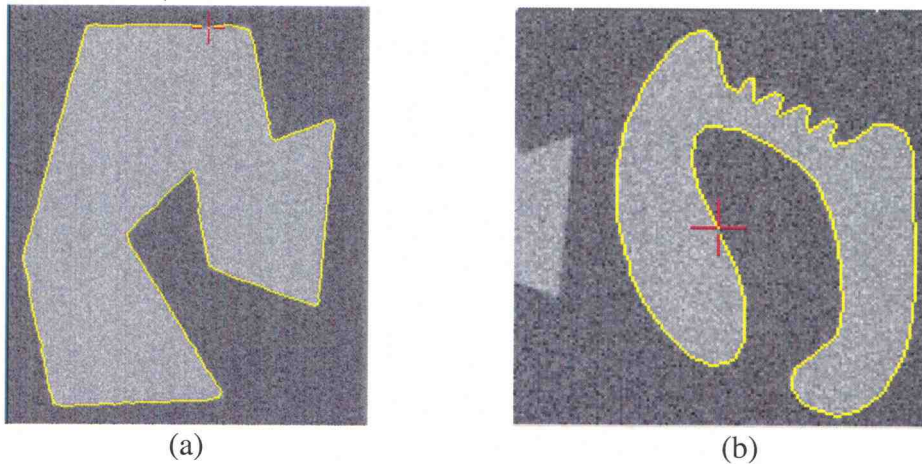


Figure 4.2 Grayscale test images with single observation. (a) Boundary of a shape with sharp corners. (b) Boundary of a shape with varying curvatures.

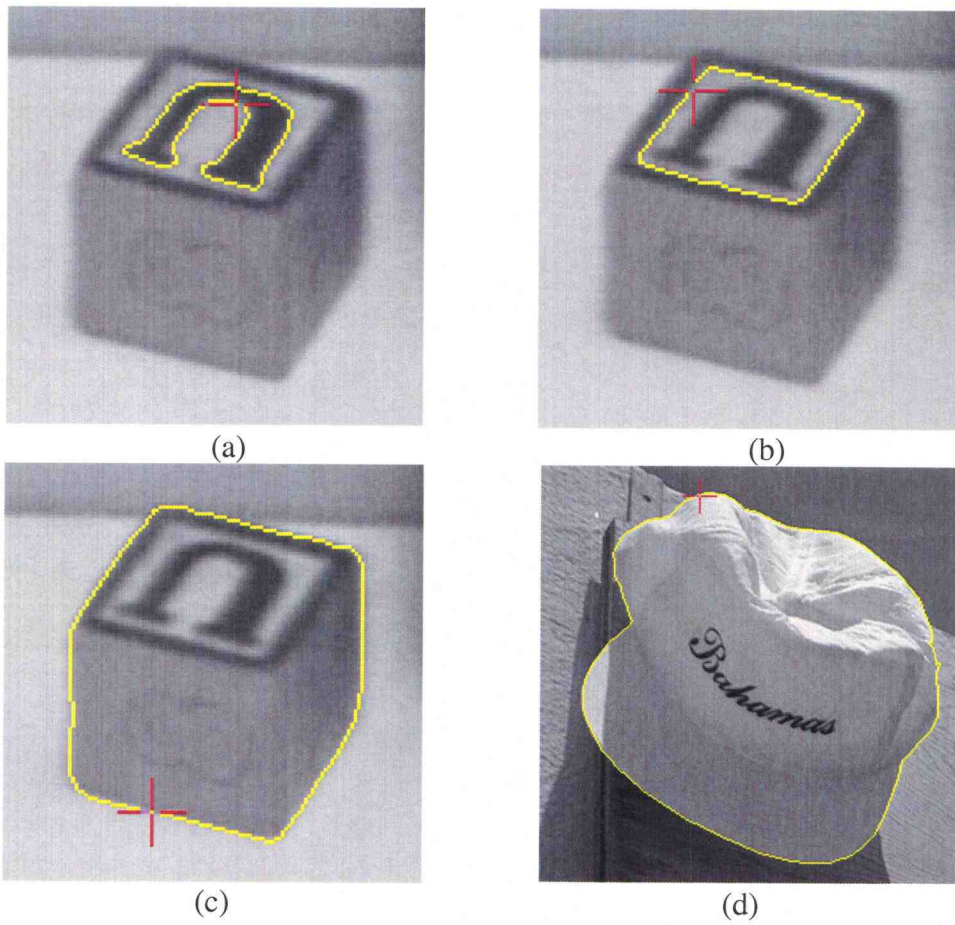


Figure 4.3 Grayscale block and hat images with single observation. (a) (b) (c) Three object boundaries in the block image. (d) Boundary of hat.

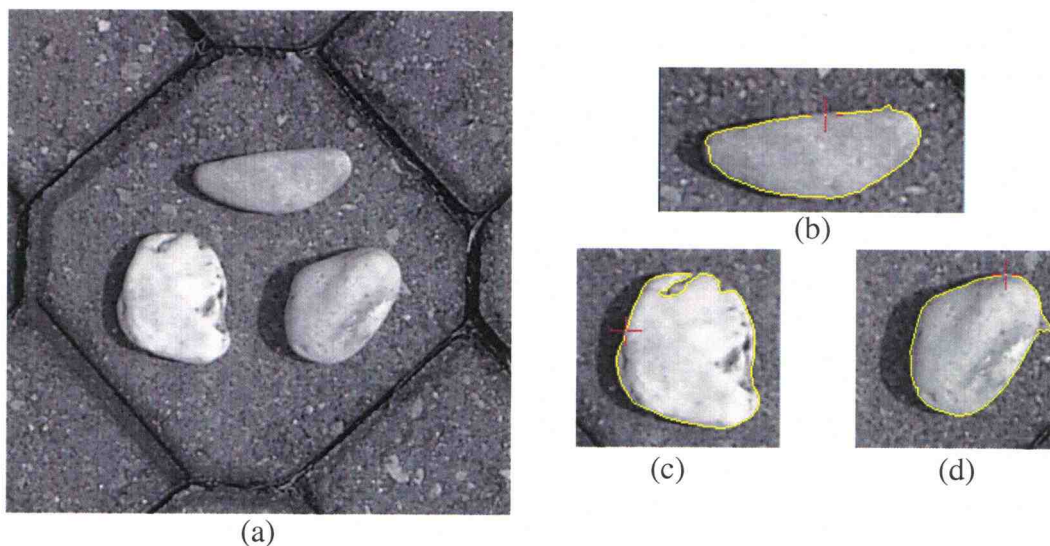


Figure 4.4: Grayscale image of stones on pavement. (a) Original image. (b) (c) (d) Boundaries of the stone images.

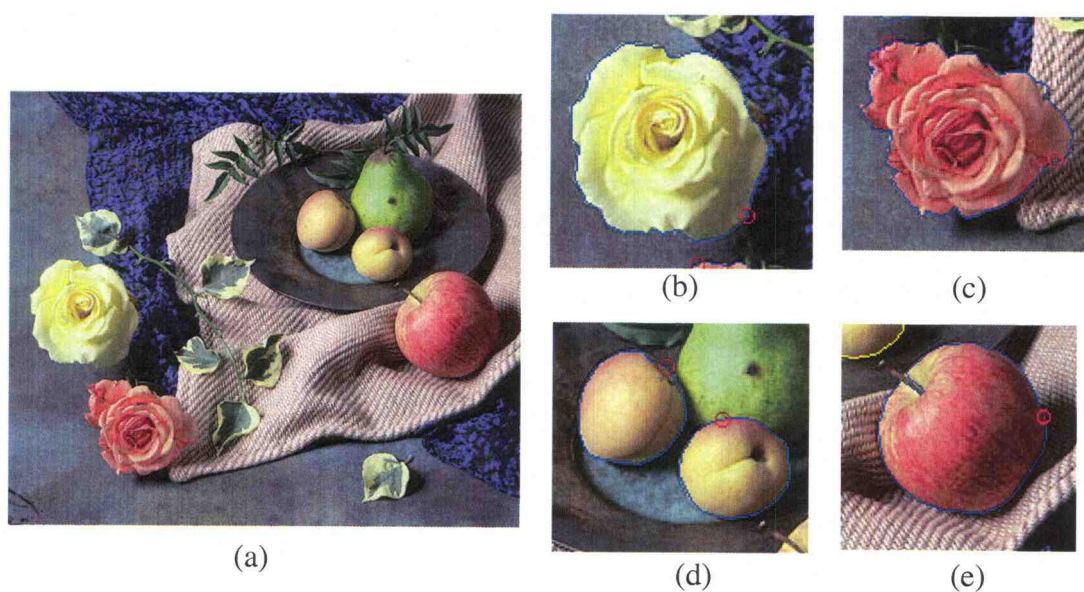


Figure 4.5: Color still life image. (b) Boundary of yellow rose. (c) Boundary of red rose, actually with two observations. (d) Boundary of apricot. (e) Boundary of apple on highly textured background.

## 4.2 Multiple Observations

The object boundaries in images of this section are not trivial and are used to demonstrate how our techniques deal with more complex object boundaries. All the object boundaries in this section are defined using multiple observations.

Let's use Figure 4.6 as an example to illustrate the process for highlighting the cow with multiple boundary observations. In Figure 4.6a there is not yet an anchored seed so the *display* seed consists of the single edge as defined by the current cursor (indicated with a white cross-hair). In this case, that single observation is sufficient to define the cow's ear, but not the entire cow. Pressing the mouse button copies the current *display* seed to the *anchor* network and computes the MPST using the new anchored seed. For each new cursor position, the minimum path from that point back to the seed path in the *anchor* MPST is combined with the *anchor* seed path to form the new seed path for the *display* network, which then recomputes and displays its MPE loop. In Figure 4.6b, the mouse has moved slightly to the left of the anchored seed edge (now shown as a blue circle) and the snapped cursor position has defined a new observation on the back of the cow. However, this is still not sufficient to define the whole cow so a third observation just below the ear is added in Figure 4.6c.

Figure 4.7 is spoon image, which is another challenging case for all current segmentation techniques. The reflection property of the spoon material produces several saturated, bright and constant intensity areas on the spoon. Humans can recognize it because we have knowledge and context information of the object. We

know what the spoon should look like and how the reflection affects the spoon, which prevents us from confusing by the reflections.

Figure 4.8 demonstrates how our techniques deal with the problem of gaps on the object boundary. Gaps are created by loss of contrast in some areas and cause problems in previous local edge following techniques in image segmentation. Our Object Highlighting solves this gap problem by providing a second observation, which creates a seed path between the first and second seed points and goes across the gap. There are still multiple gaps along the boundary of the egg. But the seed path set the scaling of the loops, other loops in or above this scaling could have more gaps and overall cost is higher than the egg's boundary loop.

Figure 4.9 demonstrates how Object Highlighting helps a normal user to edit his own images. Figure 4.9 is the upper body of human figure in an outdoor scene, which is randomly picked from a collection of image taken by a digital camera. This image has misleading background and weak edges on object boundaries. Figure 4.9a shows the boundary with the first observation and Figure 4.9b shows the human face is defined with two observations. With two observations, both the human face and neck could be highlighted showing in Figure 4.9c.

Figure 4.10 shows the sheep boundary defined with three observations. This is yet another challenging case which both the sheep and background are highly textured. There are also lighting problem on the back of the sheep and the boundary of the sheep is not smooth. Our Object Highlighting techniques could still handle this complex situation with only three observations.



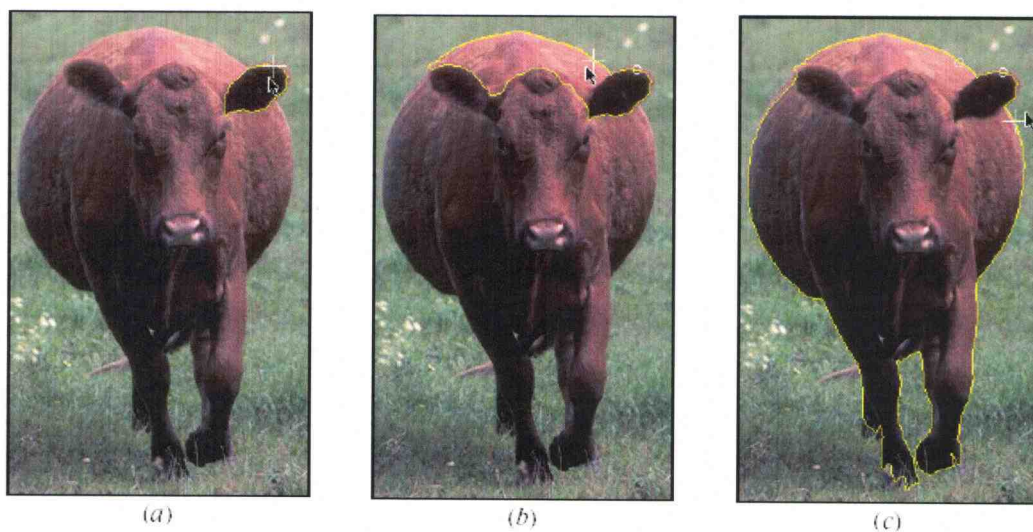


Figure 4.6: Color cow boundary defined with three observations. (a) A single observation specifies a single seed edge and is sufficient to define the cow's ear, but not the entire cow. (b) A second observation to the right of the first observation produces a seed path between the current cursor position and the first observation, but this is still not sufficient to define the entire cow. (c) A third observation augments the seed path further. There are now sufficient observations to define the cow's boundary.

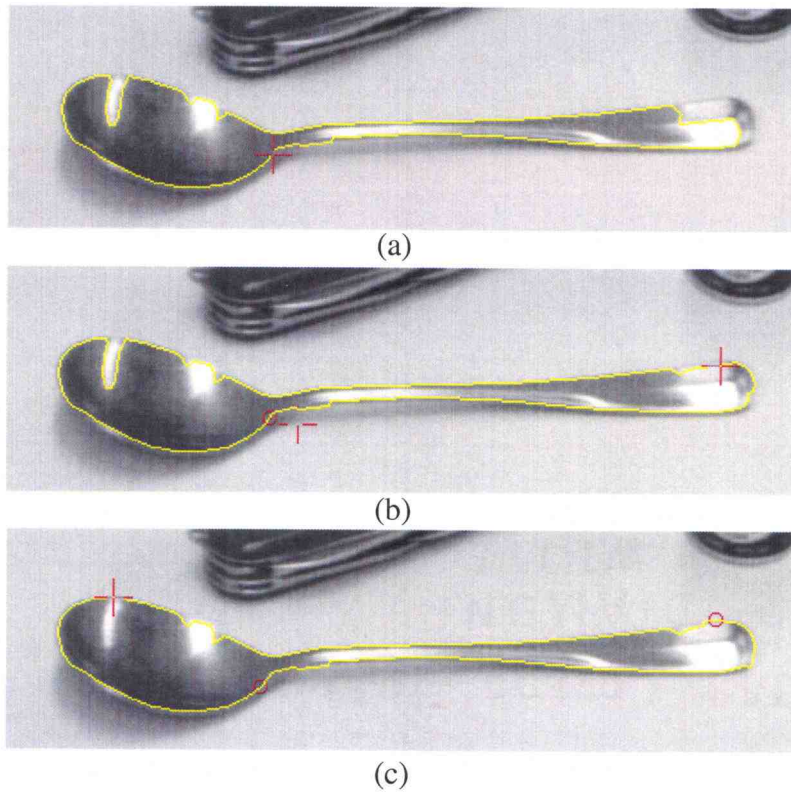


Figure 4.7: Grayscale spoon boundary defined with three observations. (a) A single observation defines most part of the spoon, but still misses two places on the right contour. (b) A second observation on the tail to fix one. (c) A third observation on the head to fix the other one. Now the spoon's boundary is mostly correct defined.

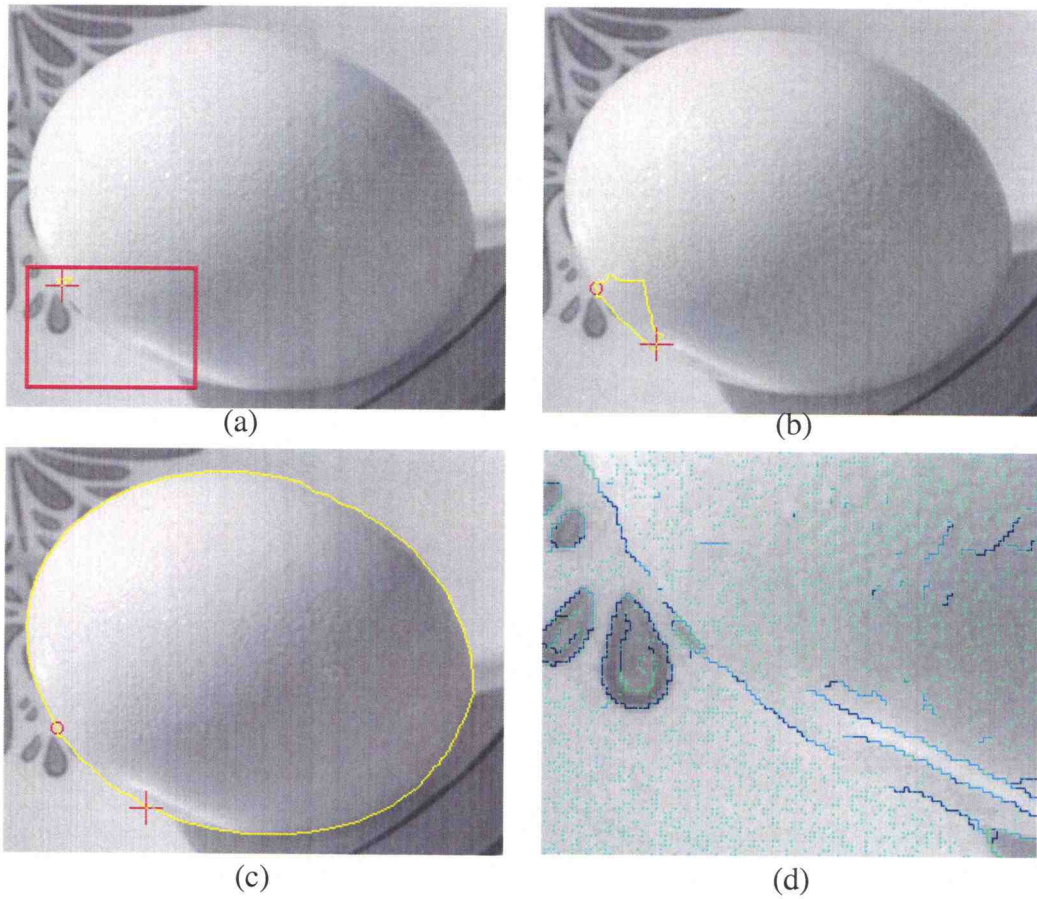


Figure 4.8: Grayscale egg boundary defined with two observations. (a)(b)(c) are the observations along the egg boundaries. (d) shows the gap problem in the zoomed area.



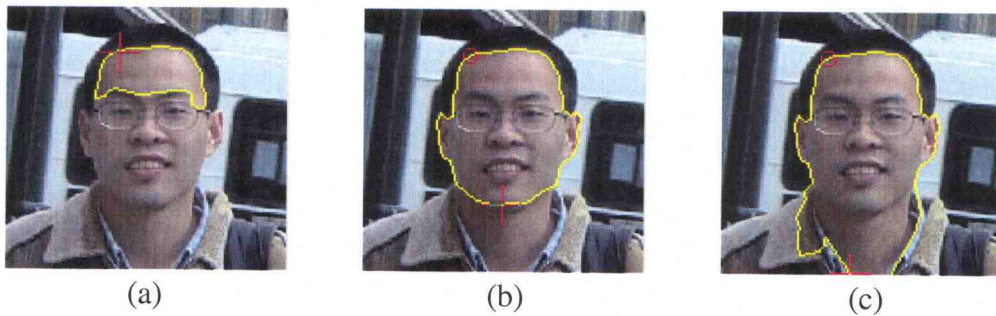


Figure 4.9 Color human face boundary defined with two observations. (a) The first observation. (b) Two observations successfully define the human face. (c) If the second observation is moved to lower part, it defines both the human face and the neck.

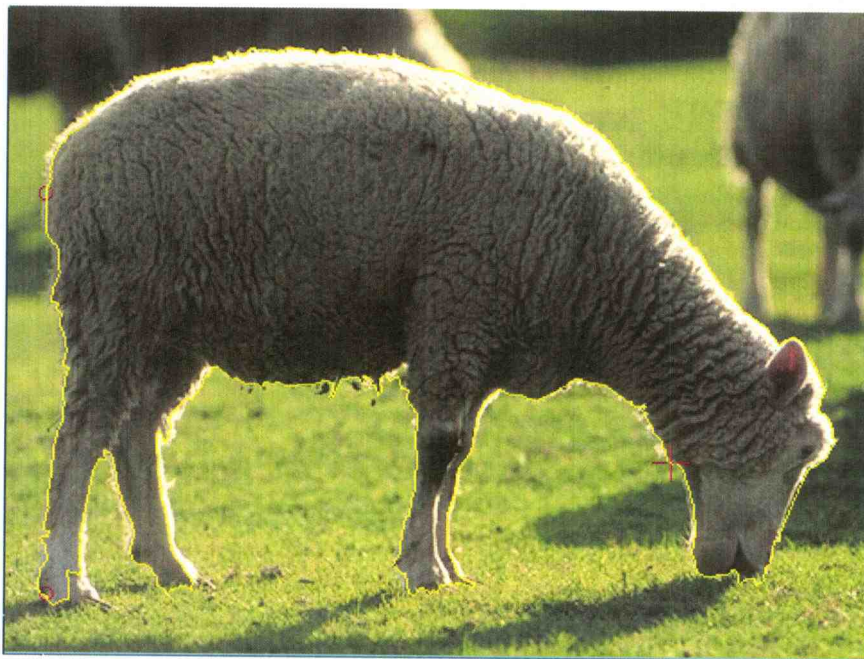


Figure 4.10 Color sheep boundary defined with three observations.

## 5. Conclusion and Future Work

### 5.1 Conclusion

In this thesis, we have introduced a Bayesian network approach that could be used for both automatic and interactive object boundary detection. We mainly focus on interactive object segmentation based on the near linear time graph search algorithm. The major contributions of this work include:

1. Use of pre-segment edges and junctions of edges as the primitives for defined variable of Bayesian network other than using the pixels, which has been used by most other researchers. This not only reduces the size of Bayesian network but also allows incorporating both edge and region based measures.

2. We present a two layer topology, which could naturally convert the planar image feature graph into our Bayesian network. This thesis also presents an innovative design of the conditional probability table of the V-nodes, constrained to find non-overlapping and non-self-intersecting closed contours.

3. The Bayesian network formulation allows both automatic and interactive object boundary detection. We mainly focus on interactive object boundary detection in this thesis. We present a near linear time algorithm for finding the most probable loop. The efficient computation allows a user to interact with the belief network to quickly define object boundaries within a general class of images. To the best of our knowledge, there doesn't seem to be any previous work that computes the most probable explanation (MPE) of a Bayesian network in near linear time under certain

assumptions and apply the results within a real-time environment that allows the user to interact with the belief network.

## 5.2 Future Work

Although Object Highlighting has dramatically decreased the time and provided flexibility for object boundary detection, there are still possible extensions of this work.

First and foremost is designing an efficient inference algorithm to compute MPE of multiple objects without evidence, which utilizes the special characteristics of our network. It should allow negative weight in the graph and also keep the length invariance features. This would make fully automatic Object Highlighting practical in some situations. As such, the current Bayesian network approach would provide a framework for both automatic and interactive image segmentation.

Second, our current probability tables mainly include low level knowledge of object boundaries, which integrates well with the Gestalt laws of proximity, continuity and closure. The prior probability of E-nodes is only local information currently. It would be helpful to incorporate some global information, like geometry shape information. This would be model based automatic Object Highlighting based on the prior knowledge of some global information. Another possible extension of the probability table is to interactively adjust the prior probability using measured distributions of boundary features from the current loop and then re-compute a new MPE loop from these learned prior probabilities.

Finally, speed up the pre-processing time for Object Highlighting technique. Right now, the image pre-processing takes several seconds for a normal sized image. The most time consuming step is the computing the confidence of all edges. The edge confidence is based the weight or cost of the edges in a weighted image graph. It would save the preprocessing time if prior probability tables of the E-nodes and V-nodes are directly computed from edge weights other than edge confidence.

## Bibliography

- [1] R. E. Bellman, "On a Routing Problem," *Quart. Appl. Math.*, 16:87-90, 1958.
- [2] S. Beucher and C. Lantuéjoul, "Use of Watersheds in Contour Detection," in *Proc. Int'l Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation*, 132:2.1-2.12, Sept. 1979.
- [3] H. Buxton and S. Gong, "Visual surveillance in a dynamic and uncertain world," *Artificial Intelligence*, vol. 78, pp. 431-459, 1995.
- [4] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679-698, Nov. 1986.
- [5] G. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, Vol. 2(2-3), pp. 393-405, 1990.
- [6] J. M. Coughlan and A. L. Yuille, "Bayesian A\* Tree Search with Expected  $O(N)$  Node Expansions: Applications to Road Tracking," *Neural Computation*. Vol. 14 (8), pp 1929-1958, August 2002.
- [7] B. D'Ambrosio, "Inference in Bayesian Networks," *AI Magazine*, Summer, 1999.
- [8] R. Dechter, "Bucket Elimination: A Unifying Framework for Probabilistic Inference" In *Uncertainty in Artificial Intelligence UAI96*, 1996, pp. 211-219.
- [9] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, 1: 269-270, 1959.
- [10] J. H. Elder and R. M. Goldberg, "Ecological statistics of Gestalt laws for the perceptual organization of contours," *Journal of Vision*, Vol. 2, No. 4, pp 324-353, 2002.
- [11] J. H. Elder, A. Krupnik and L. A. Johnston, "Contour Grouping with Prior Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 6, pp. 661-674, June 2003.
- [12] J. H. Elder and S. W. Zucker, "Computing Contour Closure," in *Proc. European Conference on Computer Vision*, Vol. I, pp. 399-412, 1996.

- [13] J. Fairfield, "Toboggan Contrast Enhancement for Contrast Segmentation," in *IEEE Proc. of the 10th International Conference on Pattern Recognition (ICPR '90)*, Vol. 1, pp. 712-716, Atlantic City, NJ, June 1990.
- [14] L. Fei-Fei, R. Fergus, and P. Perona, "A Bayesian approach to unsupervised One-Shot learning of Object categories," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2003.
- [15] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation," To appear in the *International Journal of Computer Vision*.
- [16] D. Fleet, "Bayesian Inference of Visual Motion Boundaries," in *Vision Interface (VI)*, pp. 22-26, June 2003.
- [17] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton Univ. Press, Princeton, NJ, 1962.
- [18] W.T. Freeman. "Steerable Filters and Local Analysis of Image Structure". PhD thesis, MIT Media Lab, 1992.
- [19] D. Geman and B. Jedynak, "An Active Testing Model for Tracking Roads in Satellite Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, no. 1, pp. 1-14, Jan. 1996.
- [20] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741, 1984.
- [21] H. Guo and William H. Hsu, "A Survey of Algorithms for Real-Time Bayesian Network Inference," *AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. Edmonton, 29 July 2002.
- [22] G. Guy and G. Medioni, "Inferring Global Perceptual Contours from Local Features," *International Journal of Computer Vision*, Vol. 20, pp. 113-133, 1996.
- [23] D. Heckerman, J. Breese, "Causal Independence for Probability Assessment and Inference Using Bayesian Networks," *IEEE Transactions on Systems, Man, and Cybernetics*, 26:826-831, 1996.
- [24] K. Huang and M. Henrion, "Efficient Search-Based Inference for Noisy-OR Belief Networks: TopEpsilon," *Twelfth Conference on Uncertainty in Artificial Intelligence*, Portland, OR, 325-331. 1996.

- [25] F. V. Jensen, *An Introduction to Bayesian Networks*. Springer-Verlag, New York NY, 1996.
- [26] H. Jermyn and H. Ishikawa, "Globally optimal regions and boundaries as minimum ratio cycles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, no.10, pp.1075–1088, 2001.
- [27] S. N. Kalitzin, J.J. Staal, B.M. ter Haar Romeny, M.A. Viergever, "Image segmentation and object recognition by Bayesian grouping," in *International Conference on Image Processing*, IEEE Signal Processing Society, 2000.
- [28] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *International Journal of Computer Vision*, Vol. 1, No. 4, pp. 321-331, Jan. 1988.
- [29] V. P. Kumar and U. B. Desai, "Image interpretation using Bayesian networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 18, No. 1, pp. 74-77, Jan 1996.
- [30] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their applications to expert systems," *Proceedings of the Royal Statistical Society, Series B.*, 50, 154-227,1988.
- [31] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, Vol. 43, Number 1, pp 29-44, June 2001.
- [32] Z. Li and B. D'Ambrosio, "Efficient Inference in Bayes Networks as a Combinatorial Optimization Problem," *International Journal of Approximate Reasoning*, 11, 55--81.
- [33] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu, "Segmentation of multiple salient closed contours from real images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, no. 4, pp. 433–444, 2003.
- [34] E. N. Mortensen, "Vision-Assisted Image Editing," *Computer Graphics*, Vol. 33, No. 4, pp. 55-57, Nov. 1999.
- [35] E. N. Mortensen, Simultaneous Multi-Frame Subpixel Boundary Definition using Toboggan-Based Intelligent Scissors for Image and Movie Editing, Doctoral Dissertation, Department of Computer Science, Brigham Young University, Provo, UT, Dec. 2000.

- [36] E. N. Mortensen and W. A. Barrett, "A Confidence Measure for Boundary Detection and Object Selection," in *Proc. IEEE: Computer Vision and Pattern Recognition (CVPR '01)*, Vol. I, Lihue, HI, Dec. 2001.
- [37] E. N. Mortensen and W. A. Barrett, "Interactive Segmentation with Intelligent Scissors," *Graphical Models and Image Processing*, Vol. 60, No. 5, pp. 349-384, Sept. 1998.
- [38] E. N. Mortensen and W. A. Barrett, "Toboggan-Based Intelligent Scissors with a Four Parameter Edge Model," in *Proc. IEEE: Computer Vision and Pattern Recognition (CVPR '99)*, Vol. II, pp. 452-458, Fort Collins, CO, June 1999.
- [39] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, California: Morgan Kaufman, 2nd ed., 1988
- [40] S. Mahamud, K.K. Thornber, and L.R. Williams, "Segmentation of Salient Closed Contours from Real Images," *IEEE International Conference on Computer Vision (ICCV)*, IEEE, September, 1999.
- [41] Norsys Software Corp. *Netica Manual*, 2003.
- [42] P. Parent and S. W. Zucker, "Trace Inference, Curvature Consistency, and Curve Detection," *IEEE Transactions Pattern Analysis and Machine Intelligence*, Vol. 11, No. 8, Aug. 1989, pp. 823-839.
- [43] J. Pearl. Fusion, propagation and structuring in belief networks. UCLA Computer Science Department Technical Report 850022 (R-42); *Artificial Intelligence*, Vol. 29, No. 3, 241-288, September 1986.
- [44] Rish, M. Brodie, and S. Ma, "Efficient fault diagnosis using probing, " in Proc. of 2002 AAAI Spring Symposium on "Information Refinement and Revision for Decision Making: Modeling for Diagnostics, Prognostics, and Prediction", Stanford, Palo Alto, March 25-27, 2002.
- [45] S. Sarkar and K. L. Boyer, "Integration, inference and management of spatial information using Bayesian networks: perceptual organization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 3, pp. 256-274, 1993.
- [46] E. Saund, "Finding Perceptually Closed Paths in Sketches and Drawings," *IEEE Transactions Pattern Analysis and Machine Intelligence*, V. 25, No. 4, April 2003, pp. 475-491.



- [47] Shashua and S. Ullman, "Structural saliency: The detection of globally salient structures using a locally connected network," *In International Conference on Computer Vision*, pages 321–327, 1988.
- [48] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- [49] S. E. Shimony and C. Domshlak, "Complexity of Probabilistic Reasoning in Directed-Path Singly Connected Bayes Networks," *Artificial Intelligence*, vol 151, no. 1, pp. 213-225, December 2003.
- [50] Z. Tu and S. C. Zhu, "Image Segmentation by Data-Driven Markov Chain Monte Carlo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 5, May, 2002
- [51] Z. Tu, A. Chen, A.L. Yuille and S.C. Zhu, "Image Parsing," *Proceedings of International Conference on Computer Vision. ICCV'2003*. Cannes. France.
- [52] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, no. 6, pp. 583-598, June 1991.
- [53] S. Wang, T. Kubota, J. M. Siskind, "Salient Boundary Detection using Ratio Contour," *Advances in Neural Information Processing Systems NIPS*, Vancouver, Canada, 2003.
- [54] S. Wang and J. M. Siskind, "Image segmentation with ratio cut," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, no. 6, pp. 675–690, 2003.
- [55] M. Westling, L. Davis, "Object recognition by fast hypothesis generation and reasoning about object interactions," *13th International Conference on Pattern Recognition (ICPR'96)*, Vienna, Austria, 1996
- [56] M. Westling, L. Davis, "Interpretation of complex scenes using Bayesian networks," *Asian Conference on Computer Vision (ACCV'98)*, Hong Kong, 1998
- [57] 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*, Vol. 15, no. 11, pp. 1101–1113, 1993.