IMAGE SEGMENTATION USING WATERSHED PYRAMIDS

By

ANTHONY S. WRIGHT

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1996

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1998

IMAGE SEGMENTATION USING

WATERSHED PYRAMIDS

Thesis Approved:

[illegible]

Thesis Adviser

[illegible]

[illegible]

Wayne B. Powell

Dean of the Graduate College

ACKNOWLEDGEMENTS

There are a number of people who I would like to thank for their help in completing this project. First and foremost I would like to thank my adviser Dr. Scott Acton who has always been there for me with good advice whether it be academic, professional, or personal. My thanks goes out to Andrew Segall, who always made my days in the Oklahoma Imaging Laboratory more interesting, and I wish him the best of luck on his doctorate work at Northwestern University. I would also like to thank Dr. Keith Teague and Dr. Rao Yarlagadda for taking the time to review my thesis and serve as members on my thesis committee.

Next, I would like to thank my dear friends Jeff Metcalf and Kristy Pope for helping me through a tough personal time in my life during my last few months in Stillwater at the end of 1997. I would also like to show my appreciation to another dear friend, Jennifer Leathers, who has helped to keep me motivated and maintain my sanity during the past two months as I finished writing my thesis while working full time in Tulsa. In addition, I would like to thank Mark Allen and Wayne Price, my supervisors at Williams Communications for their support and understanding. I owe the completion of this thesis to Mark's motivation, and the flexibility he allowed me in my work schedule. Finally, I have to also show my gratitude to Max, my beagle, who has been an ever-faithful companion through out my college career and is always there with a lick in the face when I need it the most.

In conclusion, I would like to thank Lucent Technologies for funding my graduate research at OSU, as well as Williams Communications for paying my education expenses for 1998.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Motivation

With applications ranging from target tracking to biomedical scanners, image segmentation is an important component of many computer vision systems in use today. Segmenting an image into homogeneous regions is the first step in most intelligent computer vision algorithms. Although the problem of dividing an image into its constituent regions appears trivial at first glance, it is one of the more complicated problems in the field of image processing. The first question that must be addressed is what constitutes a region? An image can be divided into regions based on a number of criteria. One segmentation method is to group the regions of the image where the objects in the scene have a similar texture. Another method is to segment the image based on the color or hue of different objects in the scene. A third method is to segment the image by defining regions based on the edges of objects. There are a number of other methods to segment an image, many of which will be reviewed in Chapter II.

This thesis focuses on a segmentation technique that uses the edges of objects in an image to divide the scene logically into regions. The key element of any edge based segmentation algorithm is the technique used to locate the edges, and there are a variety of techniques targeted toward finding the edges in an image. Interestingly, even though edge detection has been a major area of research for decades, there is not an accepted method that works well for every situation. The technique is almost always chosen based on compromises, and custom tailored to fit the application. Some of the typical properties of an edge detector include edge localization, sensitivity, and computational complexity. The goal of most edge detection algorithms is to find the important edges of an image quickly and accurately. The ambiguous nature of these criteria makes for some interesting questions such as: What constitutes an edge? How do you classify an algorithm as "high speed"? Exactly where is the edge of an object located? Further discussion of edge detection is found in Chapter II.

The edge detection algorithm described in this paper is based on watershed image segmentation. The watershed transform segments an image by modeling the different intensity regions of the image as topographical peaks and valleys. Watershed segmentation is a powerful tool, with many advantages over other traditional edge based segmentation algorithms. One key advantage is that the watershed guarantees thin connected edges. Another advantage is that the watershed offers good edge localization. Unfortunately due to some shortcomings of watershed image segmentation, the technique has failed to gain widespread application in the image processing community. The major downfalls of traditional watershed image analysis are that it typically results in over segmented images, and traditional techniques for determining the watershed regions are computationally intensive. Some of the traditional techniques for performing watershed image segmentation are discussed in more detail in Chapter II.

This paper introduces a method to calculate a multiresolution watershed. Multiresolution image processing algorithms introduce a sense of scale to the objects in the scene. The logic behind multiresolution processing is that not all of the information in the image is important for segmentation. The morphological pyramid technique described in this paper eliminates the smaller objects in the scene by alternately filtering and sub-sampling the image. The image pyramid works under the assumption that the smaller objects in the image are details, and that these details are not essential to segmenting the image into logical regions. The image pyramid and other multiresolution image processing methods are discussed in Chapter II.

By combining the watershed transform with the morphological pyramid, this paper addresses the two key shortcomings of the traditional watershed segmentation algorithm: over-segmentation and computational expense. The over segmentation issue is addressed by incorporating the notion of scale through the use of a pyramid. Smaller objects are eliminated when the pyramid is formed, and consequently do not appear in the resulting watershed transform of the image. The issue of computational expense is addressed since the watershed is only applied once at a coarse scale. The linking algorithm introduced in Chapter III allows the watershed to be performed once at a coarse level of the image pyramid, then the boundary information is propagated through the finer levels of the image pyramid until it reaches the original scale. Since the coarse scale representation of the image has much fewer elements than the original image, the watershed transform is not computationally prohibitive at the coarse level. Further, the linking algorithm is very simple and does not introduce a large number of computations.

In Chapter II of this paper, image segmentation and edge detection are described in detail. Chapter II also introduces multiresolution image segmentation and the image pyramid. Finally, Chapter II includes some discussion of traditional watershed segmentation algorithms, and the work done to date in the area of multiresolution watershed image analysis. Chapter III introduces the watershed pyramid and the linking algorithm used to propagate the watershed between different levels of the watershed pyramid. Chapter IV shows some visual results of applying the watershed pyramid to different images, and includes an analysis of the computational savings that comes from using the watershed pyramid instead of traditional watershed image segmentation techniques. Chapter V concludes the thesis with a summary of the watershed pyramid and some ideas for future research.

CHAPTER II

LITERATURE REVIEW OF IMAGE SEGMENTATION

2.1 Chapter Overview

This chapter offers a brief introduction to previous techniques of image segmentation, and outlines the motivation for image segmentation. The basic principles of image segmentation are discussed, and four different methods for image segmentation are presented. These four methods are edge based image segmentation, region based image segmentation, watershed image segmentation, and multiresolution image segmentation.

Section 2.2 contains a discussion of the need for image segmentation, and establishes some of the properties used to quantify an edge segmentation algorithm's performance. Section 2.3 describes some of the widely recognized edge based image segmentation techniques. The edge based segmentation techniques described include the Sobel edge detector, the Prewitt edge detector, and Laplacian based detectors. This is followed by a brief discussion of edge linking algorithms. Next, Section 2.4 overviews region based segmentation techniques, including spatial clustering, region growing, and split and merge. A brief summary of watershed image segmentation is presented in Section 2.5. The principle behind watershed image segmentation is given as well as a description of a few of the algorithms that exist to determine the watershed of a digital image, including the flooding technique, Vincent's queuing algorithm, and Meyer's queuing algorithm. Next, Section 2.5 describes multiresolution image segmentation. The image pyramid is introduced with a discussion of Burt's multiresolution segmentation algorithm using the Gaussian pyramid. Finally, Section 2.6 chapter gives a brief summary of image segmentation techniques discussed in Chapter 2 and their properties.

2.2 Principles of Image Segmentation

Segmentation plays an important role in most image processing and computer vision systems. Dividing an image into sensible regions is a logical first step in many tasks ranging from tracking to object recognition. It is segmentation that enables an image processing system to organize raw data in a manner such that processing can focus on specific regions and objects in the scene rather than the entire collection of raw data. Due to the importance of image segmentation, extensive research has been performed to develop robust segmentation algorithms. Interestingly, although image segmentation is one of the oldest areas of research in image processing, there is no one algorithm in existence that performs well in every situation. In this section, a general background of various segmentation techniques is given following a discussion of some of the properties of a well-segmented image.

One key question must be asked when designing a system to segment images. What constitutes a well-segmented image? In general, Haralick and Shapiro suggest that a well-segmented image will have the following properties (Haralick, 1992):

1. *Regions of an image segmentation should be uniform and homogeneous with respect to some characteristic, such as gray level or texture.*
2. *Region interiors should be simple and without many small holes.*
3. *Adjacent regions of a segmentation should have significantly different values with respect to the characteristic on which they are uniform.*
4. *Boundaries of each segment should be simple, not ragged, and must be spatially accurate.*

In practice, it is very difficult to satisfy these four properties simultaneously. Typically, regions of a segmented image are plagued with the problem of having small holes and ragged edges. Developing an algorithm to perform an accurate segmentation on real images is a problem that usually requires a somewhat *ad hoc* solution.

A dual of the problem of image segmentation is edge detection. At first glance it may appear that there is no difference between edge detection and image segmentation, but in practice there is a significant difference in implementation. The main distinction between edge detection and image segmentation is that most edge detection techniques do not guarantee a closed outline of the objects in the scene. Figure 2.1 shows an image segmentation that does not correspond to the edge map of the image. The human visual system can easily distinguish between the three regions of the image, but the segmentation is not based on the edges of objects in the scene.

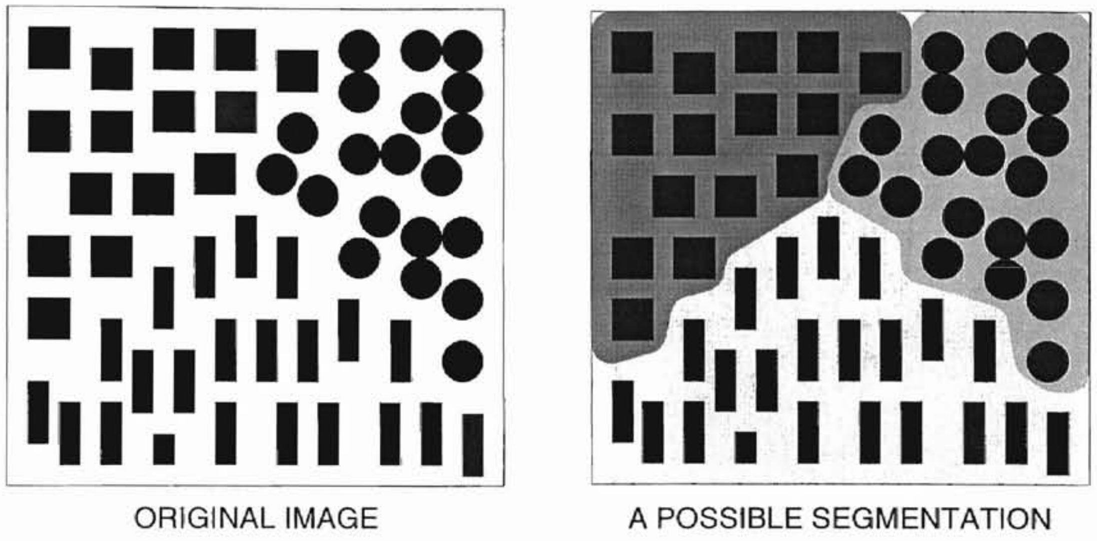

Figure 2.1 Example of image segmentation. Note that the boundaries of the segmentation do not correspond the edges of objects in the original image.

In many applications, it is desirable for the boundaries of the regions in a segmented image to correspond to edges in the original image. In such cases the segmentation forms an outline of the objects in the scene, commonly referred to as an edge map. Most edge detection algorithms do not guarantee a connected edge map, meaning that although the edge map does somewhat form an outline of the scene in the image, the scene is not divided into coherent regions. For the purposes of most computer vision and image processing systems, an edge map that does not form a meaningful segmentation has little value. An example of an edge map that does not segment the scene into coherent regions is shown in Figure 2.2.

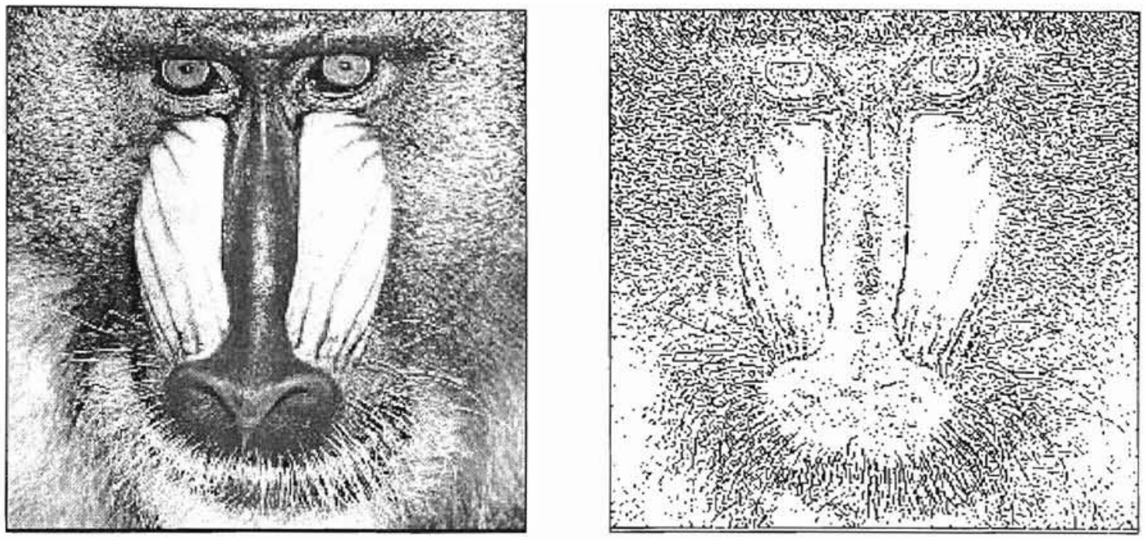

Figure 2.2 Example of an edge map. The Original image is shown on the left. The corresponding edge map is shown on the right. Note that although the edge map does form a representative "outline" of the image, it does not divide the image into distinct regions. The human visual system can identify the picture on the right, but the edge map would be of little use to most computer vision systems.

Although edge detection and segmentation are different problems, many segmentation algorithms are based on some form of edge detection. Most classic methods of segmentation fall into one of two categories: edge based segmentation and region based segmentation. The following section discusses some of the more common edge detection algorithms, and concludes with a discussion of edge linking techniques that produce image segmentation from a discontinuous edge map.

2.3 Edge Based Image Segmentation

2.3.1 Image gradient operators

The basis of every algorithm that performs edge based image segmentation is some form of edge detection. Before entering a discussion on edge detection it is important that we understand what constitutes an edge in a digital image. Gonzalez and Woods state that "an edge is the boundary between two regions with relatively distinct gray-level properties" (Gonzalez, 1993). If we assume that the gray-level intensity of each object in an image is homogeneous then an edge is any sharp discontinuity of intensity between adjacent regions.

The basis of the edge detection techniques described in this section is a local derivative operator. Figure 2.3 shows an edge and its relative derivatives. There are two basic methods used for finding the edges in an image using derivatives. One method is to calculate the first derivative of the image and then threshold the magnitude of the first derivative of each element in the image. If the magnitude of the first derivative of an element exceeds the predetermined threshold then that element is labeled as an edge. Another method relies on the second derivative of the image. Notice that in Figure 2.3 the point where the curve crosses the X-axis in the second derivative of the edge corresponds to the location of the edge in the original image. By calculating the second derivative of an image and then searching for *zero-crossings* within the result, it is possible to identify the edges within the original image.

In order to calculate differentials in a digital image space, we must first understand the definition of a gradient operator. The gradient of an image *f(x,y)* at the location *(x,y)* is the vector

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}. \qquad (2.1)$$

For the application of edge detection, it is more useful to represent Equation 2.1 in terms of the vector magnitude. From this point forward, the *gradient* will be denoted ∇f where

$$|\nabla f| = \text{mag}(\nabla \mathbf{f}) = \sqrt{G_x^2 + G_y^2}\ . \qquad (2.2)$$

An approximation of the gradient can be given as

$$|\nabla f| \approx |G_x| + |G_y|\,. \qquad (2.3)$$

Since the partial derivatives given in Equation 2.1 can be approximated as sums and differences in a two dimensional digital signal, we can calculate the derivative in image space by convolving the original image with the appropriate kernel.

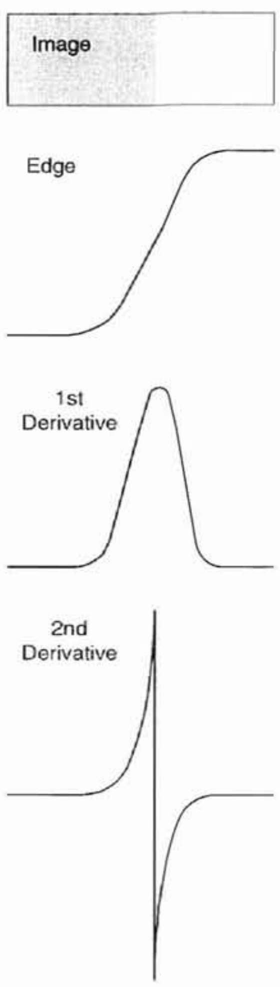

Figure 2.3 Differentiation at an edge.

2.3.2 *Sobel and Prewitt edge detectors*

One example of a gradient based edge detector is the Sobel operator (Sobel, 1970). The Sobel operator works by averaging the gradient of each element over three rows and three columns. The derivatives G_x and G_y are given as

$$\begin{aligned} G_x &= (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \\ G_y &= (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \end{aligned} \tag{2.4}$$

where the variables $z_1 \ldots z_9$ are defined as the elements of a 3x3 kernel in eight connectivity space. Figure 2.4 defines the position of each element within the kernel, and shows the coefficients of the kernels used to calculate the Sobel differentiation of a digital image based on Equation 2.4.

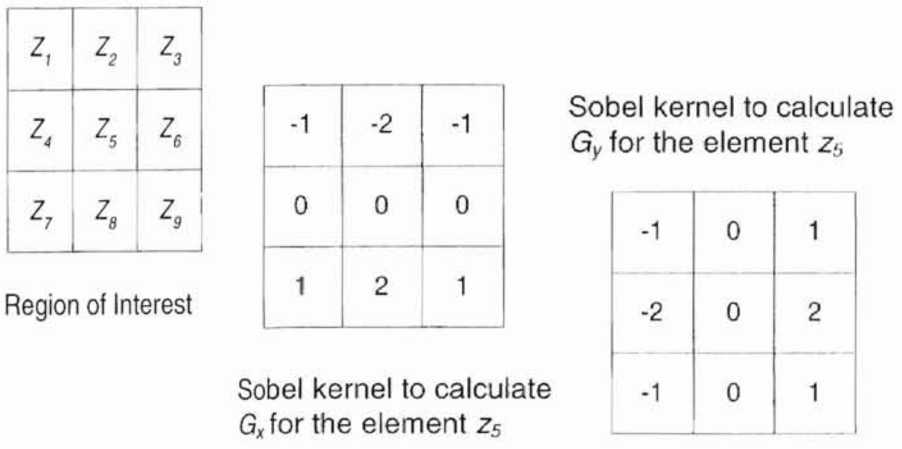
z_1 z_2 z_3
z_4 z_5 z_6
z_7 z_8 z_9
Region of Interest
-1 -2 -1
0 0 0
1 2 1
Sobel kernel to calculate G_x for the element z_5
Sobel kernel to calculate G_y for the element z_5
-1 0 1
-2 0 2
-1 0 1

Figure 2.4 Kernels for Sobel edge detection.

To find an edge map using a Sobel edge detector, the first step is to determine G_x and G_y for each element in the original image by convolving the original image with the 3x3 kernels shown in Figure 2.4. The next step is to find the gradient magnitude using the approximation given in Equation 2.3. The final step is to perform a thresholding operation and label every element whose gradient magnitude is greater than a predetermined threshold as an edge.

Another commonly used gradient based edge detector is the Prewitt edge detector. The Prewitt edge detector is similar to the Sobel edge detector in function, but uses a slightly different set of kernels to calculate the partial derivatives (Prewitt, 1970). The kernels used in the Prewitt edge detector are shown in Figure 2.5. The method for calculating an edge map using the Prewitt edge detector is identical to the method described for the Sobel edge detector with the exception of the kernels used to calculate $\partial f/\partial x$ and $\partial f/\partial y$.

Prewitt kernel to calculate G_y

-1	0	1
-1	0	1
-1	0	1

Prewitt kernel to calculate G_x

-1	-1	-1
0	0	0
1	1	1

Figure 2.5 Kernels used in the Prewitt edge detector.

An example of the Sobel edge detection algorithm is shown in Figure 2.6. Note that the result does not have connected contours, and that although reducing the threshold does result in fewer discontinuities in the edge map it also has the effect of introducing edges associated with small features such as noise.

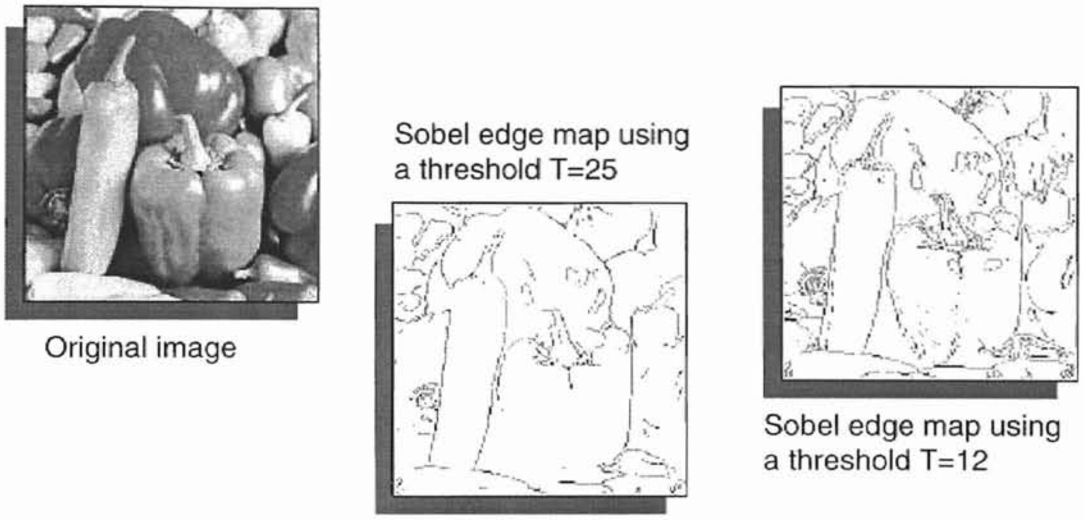
Original image
Sobel edge map using a threshold T=25
Sobel edge map using a threshold T=12

Figure 2.6 Edge detection using the Sobel edge detector. Note that as the threshold is decreased, the contours begin to close, but unwanted edges also begin to appear.

2.3.3 Laplacian based edge detectors

Another type of gradient based image segmentation relies on the Laplacian function to calculate the derivatives of the image (Gonzalez, 1993). The Laplacian of a two-dimensional function is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{2.5}$$

0	-1	0
-1	4	-1
0	-1	0

Figure 2.7 3x3 Laplacian kernel.

When Equation 2.5 is applied to a two-dimensional digital image, it assumes the form

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8). \tag{2.6}$$

The kernel used to calculate the Laplacian is shown in Figure 2.7. Since the Laplacian is based on the second derivative of the image, the result of equation 2.5 is a scalar instead of a vector, eliminating the need to find the gradient magnitude of the derivative. The edge map is found by locating the zero-crossings in the matrix that results from convolving the kernel shown in Figure 2.7 with a digital image.

A major shortcoming of using the Laplacian for edge detection is its sensitivity to noise. Marr and Hildreth propose that the image should be smoothed with a low-pass filter prior to convolving with the Laplacian (Marr, 1980). The two-dimensional Gaussian is a common choice for the pre-filter, and since convolution is associative we can combine the Laplacian and the Gaussian into a single kernel called the *Laplacian of Gaussian (LoG)* kernel:

$$\nabla^2 G(x,y) = \frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] \cdot \exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]. \tag{2.7}$$

An example using the Laplacian of Gaussian function is shown in Figure 2.8. The resulting edge map is very similar to the edge map from the Sobel edge detector shown in Figure 2.6. The Laplacian of Gaussian edge detector exhibits the same problem areas as other gradient based edge detectors. A high threshold results in few edges. The initial edge map in Figure 2.8 (σ=2, T=1.5) shows a low sensitivity to noise, but does not have closed contours. The threshold is decreased in the second edge map (σ=2, T=0.75) and the contours begin to close, but edges that correspond to small features and noise in the image also begin to appear.

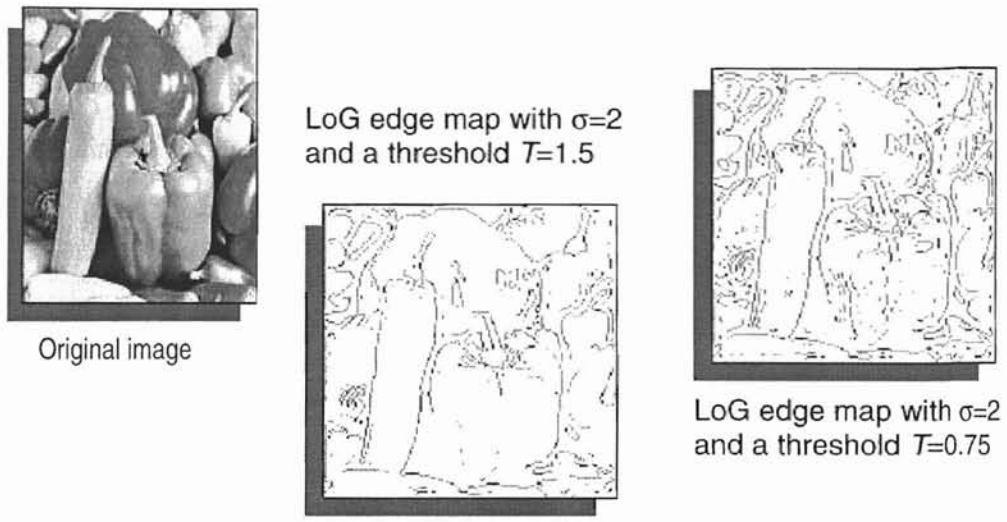
Original image
LoG edge map with σ=2 and a threshold T=1.5
LoG edge map with σ=2 and a threshold T=0.75

Figure 2.8 Edge detection using the Laplacian of Gaussian edge detector. The LoG edge detector produces a result similar to that of the Sobel edge detector. As the threshold is decreased the contours begin to close, but unwanted edges also begin to appear.

2.3.4 Edge linking algorithms

Some work has been done to close the contours that are found in discontinuous edge maps through edge linking algorithms. Most of these algorithms analyze the characteristics of elements in a small neighborhood surrounding each edge discontinuity. If similar elements are found in the local neighborhood then the elements are *linked* to form a region boundary. More complicated edge linking algorithms use nonlinear functions such as the Hough transform (Hough, 1962) to approximate the boundary between discontinuous edge points. Generally, edge linking algorithms only work if the contours in edge map are *almost* continuous, and have a tendency to create false edges.

2.4 Region Based Image Segmentation

2.4.1 Clustering algorithms

One classic method for image segmentation is the use of a clustering algorithm. In a clustering algorithm, elements which share similar features are grouped together to from regions. The similarity between regions is usually based on quantifiable image properties such as intensity, gradient, texture, and distance from other elements with similar properties. Further, the similarity measure may be based on a single image property or a weighted combination of several image properties. The variety of image properties which are available to the designer typically makes clustering algorithms a very heuristic approach to image segmentation. Typically a clustering algorithm is developed to segment a certain type of image, and performs poorly when applied to an image which differs from the type of image the algorithm was original designed to segment.An example of an application of a clustering algorithm is to segment the image using histogram mode seeking. Histogram mode seeking works under the assumption that homogeneous objects in the image appear as clusters in measurement space (Haralick, 1985). An example of spatial clustering using histogram mode seeking is shown in Figure 2.9. Although the original image appears to meet Haralick's criteria and the histogram of the image contains three distinct modes, these clusters in the measurement space do not correspond to unique objects in the image space.

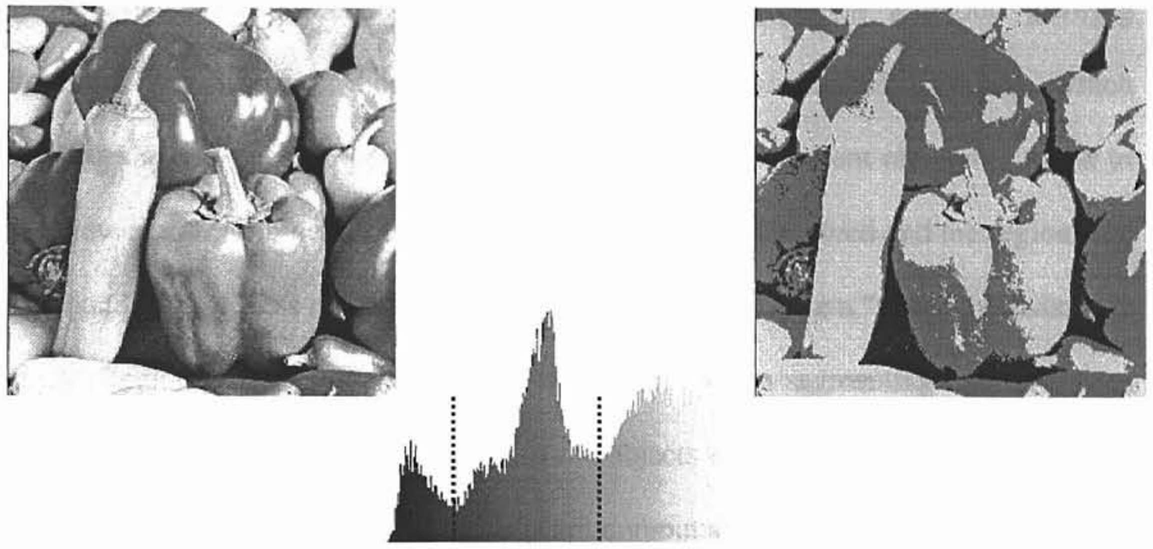

Figure 2.9 Image segmentation using histogram mode seeking. The original image is shown on the left, and the histogram of the original image is shown in the center. The image on the right is segmented into three regions based on the three major modes of the histogram.

2.4.2 Region growing algorithms

A second region based image segmentation technique is region growing. Castleman describes region growing algorithms as giving the impression that regions in the interior of objects grow until the boundaries of the regions correspond to the edges of the objects being segmented (Castleman, 1996). The first step in a region growing algorithm is to divide an image into tiny regions that may be as small as a single pixel. Each of these regions is evaluated based on properties that identify them as being part of an object in the scene. Measurement techniques vary from one implementation to the next, but properties considered might include texture or gray level intensity (Brice, 1970; Nagy, 1972; Levine, 1981). The next step is to assess all of the boundaries between adjacent regions. When designing a region growing algorithm, a measure of boundary "strength" must be assigned based on which properties are the most important. If a boundary between adjacent regions is found to be below a predetermined threshold then that boundary is dissolved and the regions are combined. The process is iterated until no adjacent regions have a "weak" boundary.

Region growing algorithms are often applied when segmenting natural scenes where little is known about the properties the objects in the image. The key weakness to region merging techniques is that they are computationally expensive and heuristic by nature.

2.4.3 Split and merge algorithms

A third popular type of region based segmentation is the split and merge method. The split and merge method is similar in function to the region growing algorithm previously discussed, but adds a *splitting* operation to *shrink* regions. The idea of using splitting algorithms was first introduced by Robertson and Klinger (Robertson, 1973; Klinger, 1973). The first step in performing image segmentation based on the split and merge algorithm is to arbitrarily segment the image. The techniques used to perform this initial segmentation vary from labeling the entire image as a single "segment", to dividing the image into N equal sections. After an initial segmentation has been determined, the split and merge algorithm does two operations. These two operations are referred to as *splitting* and *merging*.

The first operation we will discuss is the split operation, but there is no set order for the two operations. The first step is to inspect the homogeneity of each segment in the image. As with the region growing algorithm, features such as gray level intensity and texture can be evaluated to determine if the elements within a region are homogeneous (Robertson, 1973; Klinger, 1973; Horowitz, 1974). If it is determined that the members of a region are not homogeneous then the region is split into two or more regions. Like the initial segmentation, the method used for splitting regions varies from one implementation to the next. The second operation is the merge operation. This operation is identical to the region growing described previously. The boundaries between adjacent regions are inspected, and if a boundary is found to be "weak" then the two regions are merged.

The split and merge segmentation technique is very similar to the region growing technique, and suffers from the same shortcomings. The algorithms developed to implement split and merge segmentations are complicated and heuristic by nature. Typically split and merge processing must be fine-tuned to fit each application and comes with a heavy computational penalty.

2.4.4 *Other region based segmentation techniques*

There are a number of other region based segmentation algorithms in addition to the ones described in this section, but many of these region based segmentation schemes fall under the general classifications of either clustering, region growing, or split and merge. An example is the use of marker functions described by Vincent and Dougherty (Vincent, 1994). Marker function segmentation can be classified as a form of region growing. With this approach, human intervention is used to identify the objects to be segmented. After each object in the scene has been "marked", the area immediately adjacent to each marker is inspected for potential boundaries. Much like the region growing method and the split and merge method, a measurement condition for boundaries is established based on the properties of the objects in the image. Segmentation using marker functions can be simply described as a region growing technique where a human being performs the initial segmentation. Most region based segmentation methods share the same deficiencies. Region based segmentation is heuristic by nature, and typically requires costly computational time.

2.5 Watershed Image Segmentation

2.5.1 Overview of watersheds

The theory behind watershed image analysis is actually borrowed from topography. In topography, any surface can be divided into watersheds and catchment basins. Given a three dimensional topographic surface, if a drop of water were placed anywhere on the surface the water would stream down toward lower ground until it finally reached a local minimum. The set of all the points in which water drains to the same local minimum is referred to as a catchment basin. The boundaries between catchment basins are referred to as watershed boundaries. Figure 2.10 illustrates these basic concepts.The idea of using watersheds for image analysis was first introduced by Beucher and Lantuejoul in 1979 (Beucher, 1979; 1982). They introduced the notion that the watershed can be extended to digital image analysis by considering a digital image as a three-dimensional surface. In this three-dimensional representation,

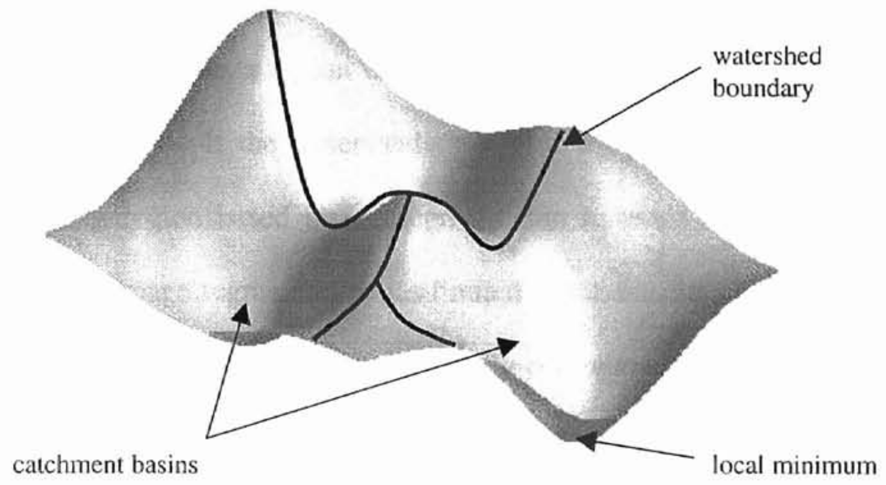

Figure 2.10 Topographical watershed. This illustrates a topographic example of local minima, catchment basins, and watershed boundaries.

elements with a high value are assigned a higher altitude than elements with a low. After this three-dimensional mapping of the image has been established, segmentation is achieved by first labeling each local minimum in the image. These local minima represent the catchment basins of the image. Next, each element is assigned to its corresponding catchment basin, resulting in a segmentation of the image. Traditional watershed segmentation algorithms are plagued with a number of shortcomings including high computational expense, and the tendency to over segment the image. Techniques to lower the computational intensity of the watershed followed quickly, the most popular of which use an immersion simulation which is described in Section 2.5.3. The immersion simulation approach simulates flooding of the image with water starting at the intensity minima, and when implemented using an ordered queue proves to be very efficient (Vincent, 1991; Meyer 1991).

One interesting feature of watershed image segmentation is that the same algorithm can act as either a region based segmentation technique or an edge based segmentation technique. If the watershed transform is applied to the original image then the resulting region based image segmentation is based on the pixel intensity. This region based image segmentation has limited use in machine vision systems since the boundaries of the segmentation do not usually correspond to the edges of objects in the image. A second form of watershed image segmentation is based on gradient edge detection. If the watershed transform is applied to the gradient magnitude of an image, the resulting image segmentation will have region boundaries corresponding to the edges of objects in the original image.

2.5.2 Steepest descent method

Following the introduction of the watershed segmentation to image processing, work began on the development of algorithms to calculate the watershed of a two dimensional digital image. Using the definition given in topography, the most intuitive way to determine the watershed of a digital image is to use the steepest descent method.

The basis of the steepest descent method is to start at each pixel in the image and trace the path a drop of water would take if the image were a three dimensional surface. This method is very effective for a continuous three-dimensional surface, but when extended to a digital surface representation with integer element values, the steepest descent algorithm is faced with two deficiencies. The first problem is the lack of an efficient way to calculate the watershed transformation. The use of a steepest descent algorithm typically requires multiple processing sweeps of the image, making the algorithms inherently computationally intensive. The second and more serious problem is how to deal with indefinite decision pixels. Since a digital image consists of discrete valued pixels, it is possible for there to be more than one path leading away from a pixel with the same "steepness", causing multiple choices for the direction of steepest descent. Steepest descent algorithms typically deal with this problem in a heuristic manner such as randomly picking one of the available paths.

2.5.3 Immersion simulations

After introducing the idea of using watersheds in digital image processing, Beucher and Lantuejoul proposed a watershed algorithm based on an immersion analogy (Beucher, 1982). Using this immersion algorithm, watersheds are computed by first assigning a distinct label to each of the local minima in the image. Since each local minima represents a catchment basin in the watershed transform of the image, each label will correspond to a distinct region in the final image segmentation. The next step is to "flood" the catchment basins of the image by performing a series of binary threshold operations on the image, starting at threshold T=0, and ending at threshold T=N, where N represents the maximum pixel value found in the image. Following each successive threshold, the regions surrounding the local minima of the image expand. When two adjacent regions merge, the pixels where the regions first made contact are defined as watershed boundaries.

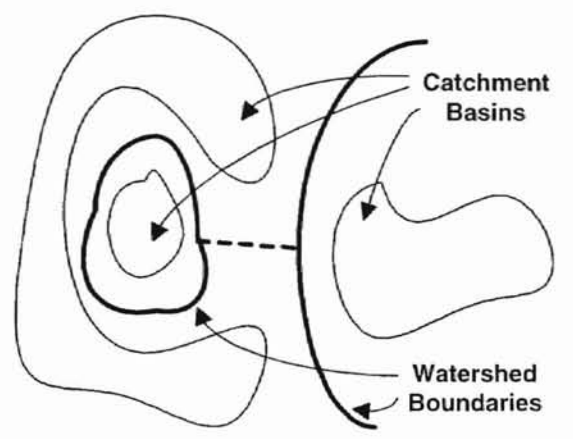

Figure 2.11 Example of a false watershed boundary using an immersion algorithm. The dashed line represents a false watershed boundary.

Although this algorithm avoids the problem of indefinite decision pixels, it offers little improvement over the steepest descent method in the way of computational efficiency, since it requires a large number of sequential scans of the image. Due to the nature of the algorithm, immersion simulations also have a tendency to produce thick watershed boundaries, which is undesirable for many applications. Another problem with Beucher and Lantuejoul's original immersion algorithm is the tendency to detect false watershed boundaries as illustrated in Figure 2.11. In this example, the dashed line is labeled as a watershed boundary, even though it does not represent a boundary between adjacent catchment basins.

2.5.4 Vincent and Meyer algorithms

Work to develop an efficient algorithm to compute the watershed transform continued, and more efficient variations of the original immersion algorithm followed. Many of these improved algorithms use an ordered queue to sort the pixels of the image, then the watershed transform is found based on these sorted pixels. The two most popular queue based watershed algorithms are the Vincent and Soille watershed algorithm and the Meyer watershed algorithm (Vincent, 1991; Meyer, 1992).

The Vincent and Soille watershed algorithm is an example of an ordered queue implementation of the watershed transform (Vincent, 1991; 1994). The first step in the Vincent and Soille algorithm is to arrange the pixels of the image in increasing order based on their gray level values. This enables direct access to the pixels at any given gray level or "elevation". The second step is to evaluate each gray level of the image from the lowest gray-level in the image to the highest gray level in the image. At each new gray level, the image is analyzed for new regional minima, and for watershed boundaries. The criteria for labeling a watershed boundary is similar to the immersion algorithm discussed in Section 2.5.3, but not every boundary between adjacent catchment basins is labeled as a watershed boundary. Instead only those pixels are equidistant from two adjacent regions are labeled as watershed boundaries. Pixels that are not equidistant from two adjacent regions, but meet the criteria for a watershed boundary given in the previous section, are given an "undefined" label. An example of an image segmentation using the Vincent and Soille watershed is shown in Figure 2.12. This image segmentation is computed by performing the watershed segmentation on the gradient magnitude of an image that had been filtered with a Gaussian kernel of variance $\sigma=3$. Due to the definition watershed boundaries used by the algorithm, the segmented image contains a number of "undefined" pixels which are shown in white.

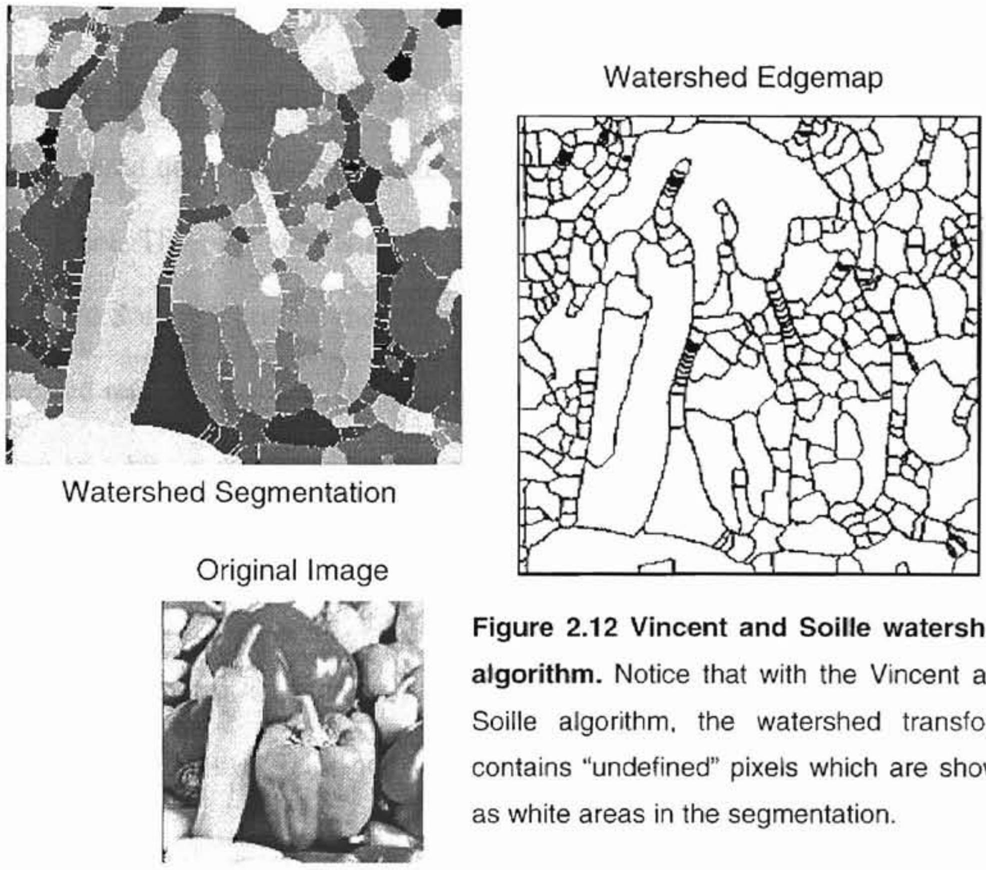

Figure 2.12 Vincent and Soille watershed algorithm. Notice that with the Vincent and Soille algorithm, the watershed transform contains "undefined" pixels which are shown as white areas in the segmentation.

The Vincent and Soille watershed algorithm offers a more efficient means of calculating the watershed transform than previous techniques. The use of an ordered queue makes the algorithm very efficient, and also avoids the problem of false watershed boundaries described in the previous section. For some applications, the presence of "undefined" elements in the image is problematic, but for such a heuristic method to assign these pixels to one of the adjacent watershed regions must be added as the final step in the watershed image segmentation. Another problem with the algorithm is its tendency to over segment the scene. This problem is shared by most implementations of the watershed, and can be overcome using region merging techniques as discussed in Section 2.4.

The Meyer watershed algorithm operates in a fashion similar to that of the Vincent and Soille algorithm (Meyer, 1990; 1992). Like Vincent and Soille, Meyer uses an ordered queue to sort the pixels of the image based on the gray level intensity of each pixel. The next step is to evaluate each gray level of the image. Unlike the Vincent and Soille algorithm, the Meyer algorithm assigns every pixel in the image to a watershed region. However, the algorithm uses a left-to-right raster scan of the image resulting in a bias towards placing pixels which would be "undefined" in the Vincent and Soille algorithm into either the region above or the region to the left of the pixel in question (Dobrin, 1994).

The algorithms described in this chapter represent some of the more common implementations of the watershed transform, but other methods include the use of grayscale skeletonization and arrowing algorithms to calculate the watershed (Beucher, 1982). One common problem of watershed algorithms proposed to date is that they are highly sensitive to variations in the gradient of an image. When the watershed transform is applied to the gradient magnitude of the image it has a tendency to over segment the original image, making it necessary to use a region merging algorithm to achieve a well segmented result. Although a number of methods have been proposed to avoid the problem of over segmentation, no one technique has been shown to be effective in every situation.

2.6 Multiresolution Image Segmentation

2.6.1 Scale space

To counteract the over segmentation problem of traditional watershed transformation algorithms, it is desirable to separate edges corresponding to important objects in the scene from those edges that correspond to noise and insignificant features. One way to categorize the edges in an image segmentation is to incorporate a sense of *scale* with each object in the scene. Objects that are large in relative size are said to be of a large *scale* while smaller features are said to be of a smaller *scale*. A collection of images from fine to coarse is commonly referred to as a scale space.

Witkin describes Gaussian convolution as a primitive scale space representation (Witkin, 1983). The Gaussian scale space described by Witkin is a sequence of images, where each successive image is calculated by convolving the original image with a Gaussian filter of increasing variance (Witkin, 1983). As the variance of the Gaussian filter increases, the insignificant features and noise begin to disappear in the original image. Figure 2.13 depicts a Gaussian scale space, represented as a three-dimensional cube where the original image is at the top of the cube and each successive image below the original has been filtered with a Gaussian filter of increasing variance. The images at the top of the cube depict fine scale representations, while the images at the bottom of the cube depict coarse scale representations.

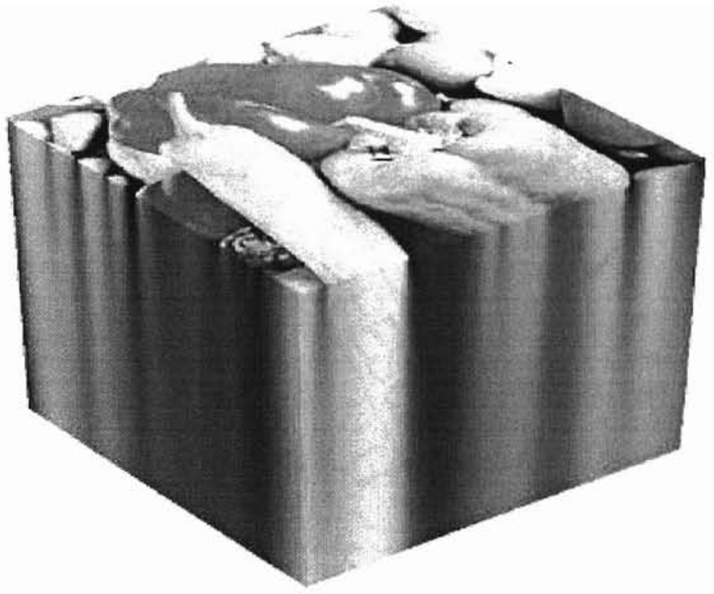

Figure 2.13 Gaussian scale space of an image. This image was obtained by successive convolutions of the original image with a Gaussian filter. The images toward the top of the cube represent a fine scale, while the images toward the bottom of the cube represent a coarse scale.

Based on this scale space representation of the image it is possible to associate a sense of scale with the edges found in the edge map of the original image. Edges that occur in a fine representation, but not in coarser representations are associated with smaller features in the scene. Figure 2.14 shows the edge map of a Gaussian scale space. This edge map was calculated by using the Laplacian of Gaussian edge detector described in section 2.3.3.

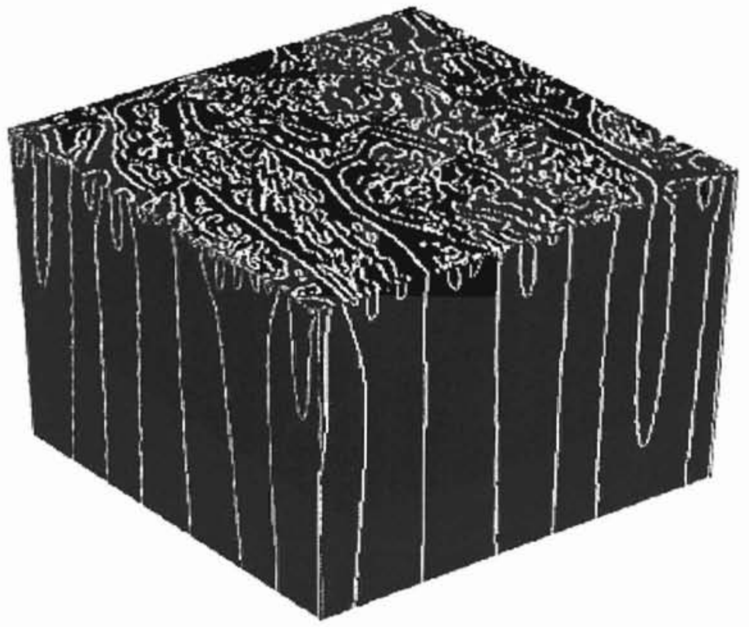

Figure 2.14 Edges corresponding to a Gaussian scale space. The edges in this figure correspond to the edges of objects in the Gaussian scale space shown in Figure 2.13. In coarser representations of the image, there are fewer edges, meaning there are fewer objects present in the scene. Also of interest is the *spatial causality* of the Gaussian scale space. Spatial causality states that no edge can exist at a coarse scale that did not exist at each of the finer scales.

An important property of the scale space edge map is *spatial causality*. The spatial causality property of the Gaussian scale space states that an edge that exists in a coarse representation of the image must also exist in each of the finer representations of the image (Witkin, 1983). Spatial causality implies that no edges can be *created* by the scale space representation, and forms the basis of a *coarse-to-fine* search. In a coarse-to-fine search, edge detection is performed at a coarse level, then the edges are "traced" through the finer levels back to the original, unfiltered image.

Although the scale space representation of an image offers a robust way to produce a hierarchy of objects based on scale, it has seen little practical application in the image processing community. This is because scale space image representations require a great deal of processing to calculate the individual levels of the scale space. In addition to computer processing time, scale spaces also require a large amount of memory to store each of the levels.

2.6.2 *Image Pyramids*

Image pyramids are a practical extension of the scale space representation of an image. Image pyramids provide a sense of scale to the objects in the scene, while reducing the total amount of information necessary to represent the image. This is accomplished by constructing a scale space as described in Section 2.6.1, and decimating the coarser representations of the image after each successive filtering operation.

Construction of an image pyramid begins by filtering the original signal. The filter is chosen to satisfy some sampling criterion, enabling the coarser representations of the image to be subsampled. Subsampling is traditionally accomplished by discarding every other row and every other column of the filtered image. This process continues iteratively until all of the objects in the subsampled image representation are at or above the desired scale. With each iteration, a new level of the pyramid is formed which is one fourth the size of the previous pyramid level. If these images are stacked with the original image at the bottom, and the coarsest representation at the top, the

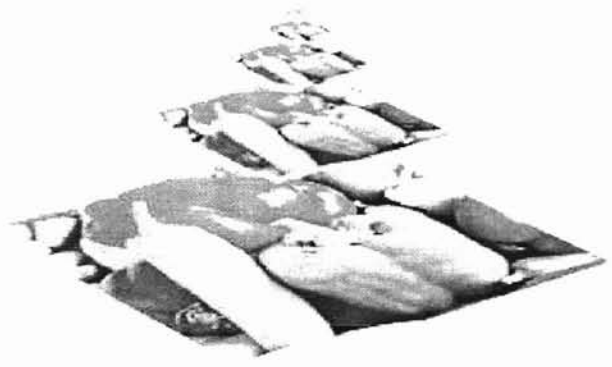

Figure 2.15 Gaussian image pyramid. This image pyramid was formed by successively filtering the original image with a Gaussian filter and subsampling by a factor of two along each row and column.

"stack" of images produces a three-dimensional pyramid. Figure 2.15 shows a pyramid image representation.

The Gaussian image pyramid shown at the left was constructed by iteratively filtering the image with a Gaussian filter and decimating each filtered result to form a more concise representation of the original image. The image at any given pyramid level L is defined as

$$\mathbf{I}_L = \left[\mathbf{G}_\sigma * \mathbf{I}_{L-1}\right]_{\downarrow 2} \tag{2.8}$$

where $\mathbf{G}_\sigma$ is a Gaussian filter of standard deviation σ (σ=2 for the image pyramid in Figure 2.15,) $\downarrow 2$ denotes a subsampling of a factor of two along each row and each column, and $\mathbf{I}_0$ is the original image (Burt, 1988).

The basis of the image pyramid is that feature extraction is performed at some level L>0 which is denoted as the *root* level of the pyramid. Since the image representation at the root level has fewer pixels than the original image, feature extraction can be done very quickly. After feature extraction is performed at this root level, the information is propagated through the finer levels of the image pyramid until it reaches the original image resolution. Although traditional image pyramids use either a Gaussian filter or a Laplacian filter, work has been done to introduce new methods of forming image pyramids. Some of these new techniques for image pyramids include the morphological image pyramid and the anisotropic diffusion pyramid (Eichmann, 1988; Acton, 1994).

2.6.3 Edge Detection Using the Gaussian Pyramid

An application of the Gaussian pyramid is to perform a hierarchical edge detection using a coarse-to-fine edge search. An is shown in Figure 2.16.

The first step of the pyramid edge detection is to construct the Gaussian pyramid as described in Equation 2.8. The next step is to perform edge detection at the *root* level of the image pyramid. For the example in Figure 2.16, a gradient based edge detector was used, and the root level is defined as pyramid level 3. The final step is to link the edge information from the root level to the original image. To accomplish this edge linking, a simple algorithm was used that performs edge detection at every level of the image pyramid. After this initial edge detection, a coarse to fine search is done to determine which edges in the original image correspond to edges in the root level of the pyramid. All edges in the original image which do not directly correspond to edges in the root level edge detection are discarded, leaving only the edges that correspond to large scale objects in the original scene.Pyramid based feature extraction does present some problems in a practical application. The edge detection shown in Figure 2.16 exhibits some undesirable features such as discontinuous contours and edges that correspond to noise and unwanted features. One remedy for the problem of discontinuous region boundaries is to use a more complex edge linking algorithm to trace edges through the coarse-to-fine hierarchy. Another solution is to use an edge linking operation as described in Section 2.3.4 to close the contours in the resulting edge detection. The problem of unwanted edges resulting from insignificant features can be handled by performing additional processing on the root level edge detection before linking it to the original image. Despite these drawbacks, multiresolution image processing architectures such as the image pyramid offer some attractive advantages over single resolution image processing techniques. The pyramid incorporates a sense of scale with the objects in the scene while reducing the processing time required to perform the initial feature extraction.

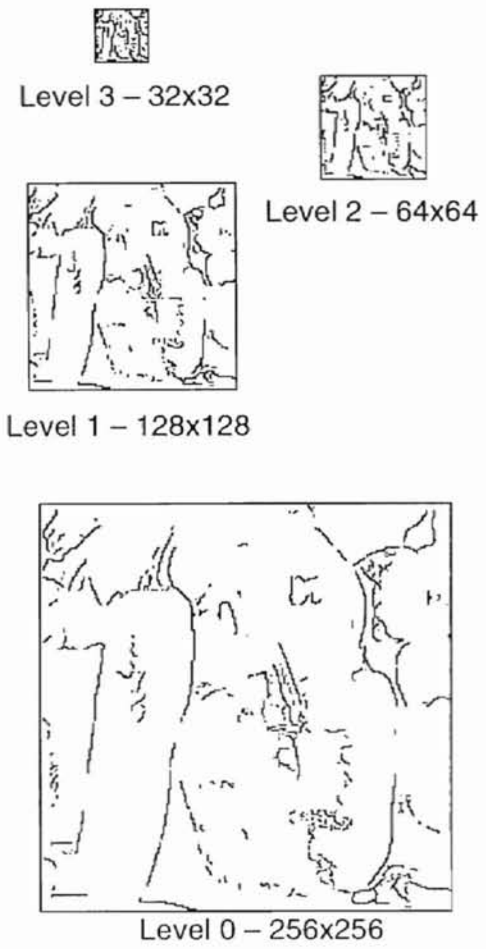

Figure 2.16 Edge detection using a Gaussian pyramid. In this example, the Gaussian pyramid was formed using a Gaussian filter with σ=2. A gradient based edge detector was used to perform edge detection at level 3, and the information was linked back to level 0.

2.7 Chapter Summary

Within the image processing community, there has long been a demand for a robust algorithm to segment digital images Although there is a wide variety of techniques to segment images into logical regions, there is no one algorithm that is robust for every application. Each image segmentation method described in this chapter has application in segmenting certain images, but each method also has some shortcomings. Although they are computationally efficient, gradient based image segmentation algorithms typically result in either over-segmented results, or discontinuous contours. Region based image segmentation algorithms guarantee closed contours, but are computationally expensive and almost always require fine tuning for each specific application. Watershed based image segmentation also guarantee closed contours and can be implemented efficiently using ordered queues, but almost always result in over segmentation. Multiresolution image processing techniques such as the image pyramid offer a way to reduce the computational complexity of many image processing algorithms while also incorporating a sense of scale with the edges in the resulting image segmentation. However, most of these algorithms suffer from ineffective linking algorithms to correlate results found in the coarse scale image to the original image.

Research continues in the image processing community to develop an image segmentation algorithm that performs well for a wide variety of images. The image segmentation algorithm presented in Chapter 3 offers an alternative solution to the traditional segmentation algorithms developed to date. This new technique uses a pyramid based watershed algorithm which incorporates a sense of scale into the image segmentation, guarantees closed contours, and is computationally efficient.

CHAPTER III

WATERSHED PYRAMIDS

3.1 Chapter Overview

Watershed image segmentation is a powerful tool for segmenting images into homogeneous regions based on the edges of objects in the image. Certain properties of the watershed function make it well suited for image segmentation. One of these properties is that when applied to the gradient magnitude of an image, the region boundaries in the resulting watershed image segmentation correspond to the edges of objects in the original scene. Another attractive property of watershed segmentation is that the resulting image segmentation is guaranteed to have closed contours. This property is particularly important for many computer vision algorithms such as object recognition.

Although watershed segmentation has a number of features which make it attractive for image segmentation, most traditional watershed algorithms also exhibit a number of undesirable properties. As discussed in Chapter 2, many traditional watershed transforms are computationally expensive, making them unacceptable for real-time applications. Another problematic property of traditional watershed segmentation algorithms is the tendency to over-segment the image. The development of queue based immersion algorithms such as the Vincent and Soille algorithm (Vincent, 1991) and the Meyer algorithm (Meyer, 1992) has helped to remedy the computational issues surrounding the watershed transform, but these methods still over-segment most images. Queue based algorithms also tend to result in watershed boundaries that are multiple pixels in width, a property that is undesirable for many applications. Traditionally, over segmentation has been remedied by either incorporating a heuristic threshold into the watershed algorithm or applying a region merging operation to the resulting image segmentation.

In this chapter an efficient algorithm to perform watershed image segmentation is presented. This algorithm avoids over-segmentation by incorporating a sense of scale with the objects in the resulting image segmentation through the use of an image pyramid. The image pyramid also results in increased computational efficiency over traditional algorithms. Finally, based on the definition of watershed segmentation presented in this thesis the resulting edge map is guaranteed to have boundaries that are of a single pixel width.

In Section 3.2 mathematical definitions are given that form the basis of watershed image segmentation. Section 3.3 describes Gauch's steepest descent algorithm, which is an integral part of the watershed pyramid. Next, Section 3.4 describes the morphological pyramid image structure used in this research. Some basic principles of morphology are discussed, as well as properties of the morphological pyramid. Section 3.5 introduces the watershed pyramid and the steps used to construct the pyramid. Finally, Section 3.6 concludes this chapter with a summary of the watershed pyramid and its fundamental properties.

3.2 Watershed Definitions

The extension of watersheds to a two dimensional digital image space requires some basic mathematical definitions. The basic premise of the watershed transform was discussed in Section 2.5.1. In this section, formal definitions are given for a local minimum, a catchment basin, and a watershed boundary. These definitions are based on the watershed algorithm used in this research, and thus differ slightly from the definitions given by previous authors such as Vincent and Meyer.

The first step is to clearly define a two dimensional digital image space. Let D_I, where $D_I \subset \mathbf{Z}^2 x \mathbf{Z}^2$, denote the domain of a two-dimensional digital gray scale image **I** which is based on a square grid of eight connectivity. The range of **I** is the set $R = \{0,1 \ldots K\}$, with K a positive integer. We also denote $N(p)$ to be the local neighborhood of the element $I(p)$. In other words, $N(p)$ is set of all elements adjacent to $I(p)$ in eight-connectivity.

A *path* P of length l between two pixels p and q in the image **I** is an $(l+1)$-tuple of elements $(p_0, p_1, \ldots, p_{l-1}, p_l)$ such that $p_0 = p$, $p_l = q$, $p_{i+1} \in N(p_i)$, and $\forall i \in \{1,2,\ldots,l\}$, $(p_{i-1}, p_i) \in D_I$.

A *local minimum* M of an image **I** is a connected plateau of pixels having the altitude h from which there is no path to an element with a lower altitude than h, that does not include at least one element with an altitude greater than h. This implies that

$\forall p \in M$, $I(p) = h$. This also implies that $\forall q \notin M$ and $\forall p \in M$ such that $I(q) \le I(p)$, it holds true that $\forall P = (p_0, p_1, \ldots, p_l)$ connecting $p = p_0$ and $q = p_l$, $\exists i \in \{1,2,\ldots,l-1\}$ such that $I(p_i) > I(p_0) = h$.

The *path of steepest descent* is a path $P = (p_0, p_1, \ldots, p_l)$ where $\forall i \in \{1,2,\ldots,l\}$, $I(p_i) \le I(p_{i-1})$, and $\forall p' \in N(p_{i-1}), I(p_i) \le I(p')$. In other words the altitude of every element p_i in the path P is less than or equal to the altitude of every element in the local neighborhood of the element p_{i-1}. In a digital image space, every path of steepest descent ends at a local minimum M.

A *catchment basin*, denoted C_M, is defined as the set of all elements in an image whose steepest descent path ends at the local minimum M. By definition each catchment basin contains only one local minimum. It can be shown that every element in the image I is a member of one and only one catchment basin C_M.

A *watershed boundary* W is defined as an element d in the image $\mathbf{I}$ such that $d \in C_M$, and $\exists d' \in N(d)$ such that $d' \notin C_M$. In other words, the element d is a member of the catchment basin C_M, and has a neighboring element d' that is *not* a member of the catchment basin C_M.

3.3 Gauch's Steepest Descent Algorithm

3.3.1 Mathematical description

For the purpose of this research, a method introduced by Gauch and Pizer (1993) is used to calculate the watershed regions. Gauch and Pizer's algorithm is based on the steepest descent method of calculating the watershed image segmentation. Despite the shortcomings of steepest descent algorithms discussed in Section 2.5, this algorithm was selected because it extends nicely to a pyramid image structure.

The first step in calculating the watershed image segmentation using this algorithm is to convert the digital image to a real number representation. This step is necessary to avoid the problems associated with ambiguous descent paths encountered in traditional implementations of steepest descent watershed segmentation. Once the image has been converted to a real number representation, it is blurred with a low variance Gaussian (Gauch, 1993). This filtering of the image has two effects. The first effect is to help reduce the likelihood of indefinite decision pixels in the image. The second effect is to help smooth over noise and other small scale features that are unimportant.

Given an original image **I** and a low variance Gaussian kernel **G**, the blurred floating point representation **B** becomes

$$\mathbf{B} = \mathbf{G} * \mathbf{I}. \qquad (3.1)$$

The smoothing of the Gaussian filter helps to improve the approximation of a three-dimensional surface provided by a discrete valued digital image, and helps to insure that each element of the image will be unique in a local neighborhood **N** (Gauch, 1993).

After the image has been converted to a real number representation, the next step is to locate all of the local minima of **B** as defined in Section 3.2. Each of these local minima is given a unique label, so that the set of all local minima contained in the image **B** is denoted as $M=\{M_1, M_2,\ldots, M_n\}$ where n is the number of local minima contained in the image **B**. By definition, each catchment basin C_M in the image **B** contains one and only one local minimum M_i, so that the set of all catchment basins C is denoted as $C=\{C_1, C_2,\dots,C_n\}$. It follows that $\forall i \in \{1,2,..n\}$ $\forall j \in \{1,2,..n\}$, $i \neq j$ $M_i \subset C_i$ and $M_i \not\subset C_j$.

After each local minimum has been located and labeled, the final step in evaluating the watershed transform is to assign each element of **B** to the appropriate watershed region. This is accomplished by starting at each element in **B** tracing the path of steepest descent as defined in Section 3.2. Each element d in the image **B** assumes the label i, where i corresponds to the label of the local minimum M_i that is found at the end of the path of steepest descent leading away from element d. As stated before, the local minimum M_i is a part of the catchment basin C_i, and based on the definition given in Section 3.2, it follows that the element d is also a member of the catchment basin C_i. It also follows that given a path of steepest descent $P=(p_0, p_1,\dots, p_l)$ where $p_0 = d$, and $p_l = M_i$, $\forall j \in \{0,1,2,\dots,l\}, p_j \in C_i$. This implies that given a path of steepest descent P, if $p_0 \in C_i$, then $P \subseteq C_i$. In other words, if an element d is a member of the catchment basin C_i, then each element in the path of steepest descent from d_0 to the minimum M_i is also a member of the catchment basin C_i. An example of the path of steepest descent is shown in Figure 3.1.

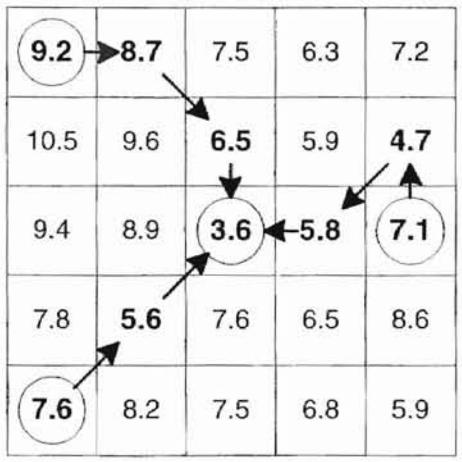

Figure 3.1 – Path of steepest descent. This is an example of tracing the path of steepest descent away from an element in a digital image. In each case, an open circle denotes the starting element and a shaded circle denotes the local minimum. Note that each intermediate element in the path of steepest descent corresponds to the same catchment basin as first element in the path.

Another useful property of the path of steepest descent is that given two paths of steepest descent $P' = (p'_0, p'_1, ..., p'_k)$ and $P = (p_0, p_1, ..., p_l)$ if $\exists q$, $(q \in P \text{ and } q \in P')$, then $p'_k = p_l$. It follows by the definition of a path of steepest descent given in Section 3.2 that if $P \subseteq C_i$, then $P' \subseteq C_i$. In other words if two paths of steepest descent, P and P' intersect, then every element that is a member of the path P and every element that is a member of the path P' is a member of the same catchment basin C_i.

3.3.2 Application to a digital image

Using properties of a path of steepest descent, watershed segmentation can be performed using two sequential passes through the image based on Gauch and Pizer's algorithm. In the first pass, every pixel *d* in the image **I** is compared to every other pixel in the neighborhood *N(d)*, and the direction of the lowest valued pixel in the neighborhood is recorded in the watershed image **W**. The range of **W** is defined as $R = \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}, \mathrm{NE}, \mathrm{NW}, \mathrm{SE}, \mathrm{SW}, M_1, M_2, ..., M_n\}$ where N represents a direction of steepest descent to the "north" of the element *d*, S represents a direction of steepest descent to the "south" of the element *d*, and so on. If the element *d* is a part of a local minimum as defined in Section 3.2, *W(d)* is assigned the value M_j, where M_j represents the label of the j^{th} unique local minimum in the image **I**.

In the second pass through the image, the path of steepest descent leading away from each element is traced to a local minimum, and each element in the image **W** is assigned to a catchment basin. Beginning at an element *d*, we follow the path of steepest descent until it either ends at a local minimum M_i, or intersects with another path of steepest descent that has been previously labeled. The first case occurs when the path ends at a local minimum M_i. For this case, the label *i* is given to every pixel that is a member of the path of steepest descent from *d* to M_i, where *i* corresponds to the label of the catchment basin C_i which contains the local minimum M_i. The second case occurs when the path of steepest descent intersects with an element d' that is not a local minimum, but has already been assigned to a catchment basin. In this case, every element in the path of steepest descent is assigned the label *i*, where *i* corresponds to the label of the catchment basin C_i of which d' is a member. This due to the property that if two paths of steepest descent, P and P' intersect, then every element that is a member of the path P and every element that is a member of the path P' is a member of the same catchment basin C_i. Since the element d' has already been assigned to a catchment basin C_i, d' is a member of a path of steepest descent P' that extends from d' to M_i. It follows that since the path P intersects with the element d' then every element in the path P is a member of the catchment basin C_i.

Once each element in **B** has been assigned a watershed region, the watershed transform is complete. From this point on the watershed transform will be denoted WS so the watershed **W** is defined as

$$\mathbf{W} = \mathrm{WS}(\mathbf{B}) = \mathrm{WS}(\mathbf{G} * \mathbf{I}) \tag{3.2}$$

In order to use the watershed function described in the previous section to find edges, the WS function must be applied to a blurred gradient representation of the image. Thus the watershed of the gradient becomes

$$\mathbf{W} = \mathrm{WS}(\mathbf{G} * |\nabla \mathbf{I}|) \tag{3.3}$$

where $\nabla\mathbf{I}$ represents the gradient magnitude of the image **I** as defined in Equation 2.2.

By applying the watershed transform to the gradient of the image, the edges are defined as the boundaries between adjacent watershed regions. The resulting edge map **E** is defined as

$$E(d) = \begin{cases} 1 & \nabla W(d) > 0 \\ 0 & otw. \end{cases} \tag{3.4}$$

An example of Gauch's watershed algorithm is shown in Figure 3.2. In this example, the image is pre-filtered with a Gaussian filter with $\sigma = 3$. Gauch's watershed algorithm was then applied to the gradient magnitude of the pre-filtered image. The resulting image segmentation contains region boundaries that correspond to the edges of objects in the original scene, but even with a high variance Gaussian pre-filter the image is over-segmented.

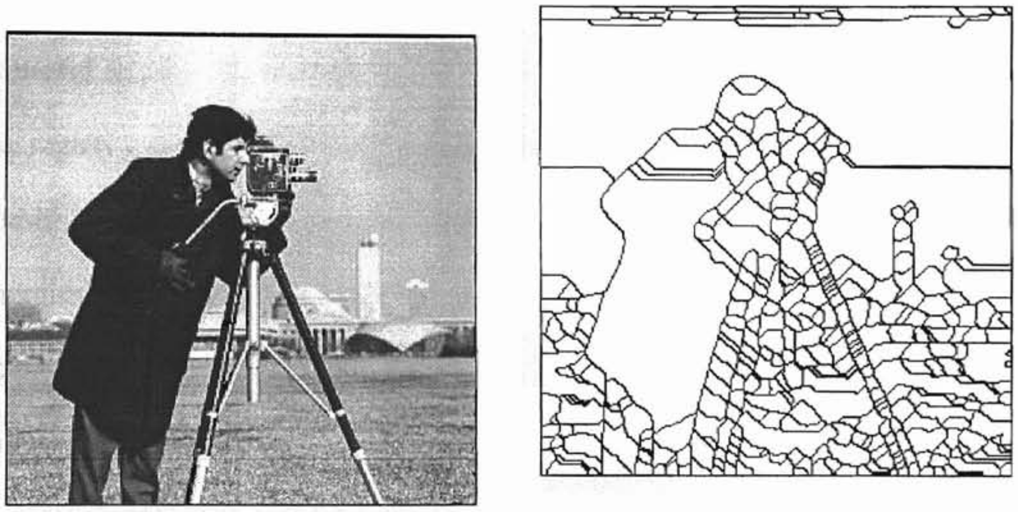

Figure 3.2 Example of Gauch's steepest descent watershed algorithm. The watershed segmentation on the right was calculated by applying Gauch's algorithm to the gradient magnitude of the pre-filtered image. In this case, the image was pre-filtered with a Gaussian filter with $\sigma = 3$.

3.3.3 Region merging

Although Gauch's algorithm is effective in segmenting digital images, a key shortcoming of the technique is the tendency for over segmentation. There are a number of methods to merge watersheds and remedy the over-segmentation problem in the literature, but for the purposes of this research, region adjacency graph (RAG) processing is used to combine watersheds (Saarinen, 1994). After the image is segmented into watershed regions, an RAG is formed from the segmented watershed based on spatial adjacencies. Two regions are said to be adjacent if they share a common boundary. The RAG also contains user defined information such as the average pixel intensity of a region, the variance of the pixel intensity of a region, the length of the boundary shared by two adjacent regions, or any other information which is useful in evaluating which regions should be combined. Much work has been done on region merging (Koshimizu, 1998; Vincent, 1994), but for the examples in this thesis a simple RAG algorithm based on mean pixel intensity is used for region merging. The first step in this procedure is to assign each watershed label a pixel intensity representative of the original image. The array of mean pixel intensities X is defined as

$$X(w) = \frac{\sum_{(d)\in \mathbf{WS}_w} I(d)}{|\mathbf{WS}_w|} \tag{3.5}$$

where $\mathbf{WS}_w \equiv \{(d) \Rightarrow W(d) = w\}$, and $|\mathbf{WS}_w|$ is the cardinality of $\mathbf{WS}_w$. X has the range $R = \{w_1, w_2, ..., w_n\}$ where n is the number of regions in the watershed image $\mathbf{W}$.

Next, adjacent watersheds in $\mathbf{W}$ are merged based on the representative pixel intensities $\mathbf{X}$ found by equation 3.5. The watershed $\mathbf{W}$ is modified by merging adjacent regions with a difference in mean pixel intensity less than a predetermined threshold T. The modified watershed $\mathbf{W'}$ is then defined as

$$W'(d) = \begin{cases} w_i & |X(w_j) - X(w_i)| < T \\ w_j & otw. \end{cases}, \tag{3.6}$$

$\forall w_i \in \text{Adj}(w_j)$, where $w_j = W(d)$ and $\text{Adj}(w_j)$ represents the set of all regions adjacent to the region w_j. An example of this region merging technique is shown in Figure 3.3. In this example, the RAG region merging algorithm described above is applied to the watershed image segmentation shown in Figure 3. The region merging algorithm has the desired effect of simplifying the image segmentation, but the algorithm also has a high computational expense and still leaves the image somewhat over-segmented.

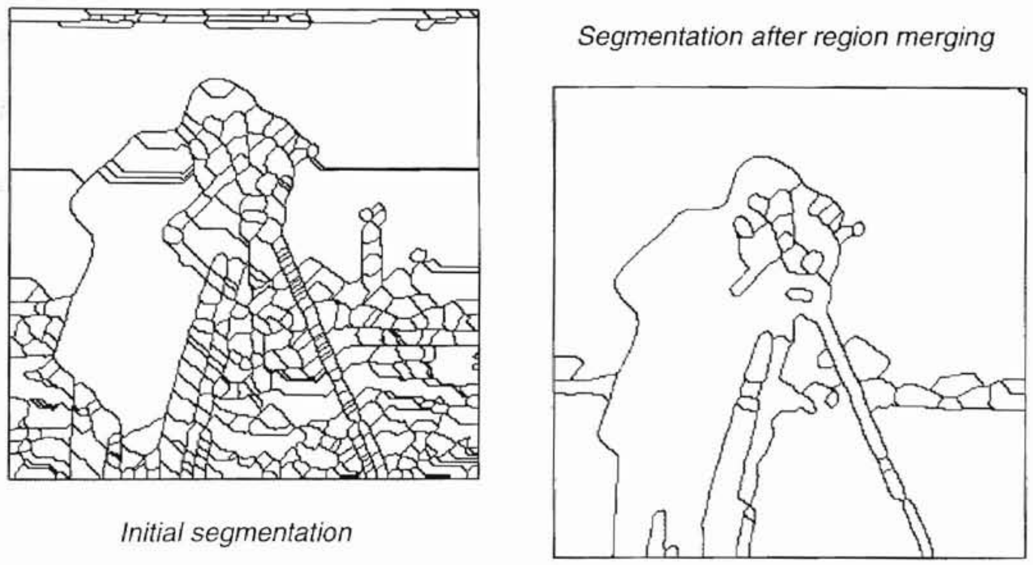

Figure 3.3 Example of RAG based region merging. In this example, the original image segmentation was calculated using Gauch's watershed algorithm after pre-filtering the original image with a Gaussian filter with $\sigma = 3$. The region merging was performed based on a threshold $T = 24$.

3.4 Watershed pyramids

As discussed in Chapter 2, an image pyramid is a practical form of a digital image scale-space. Image pyramids are formed by filtering and decimating the original image, creating a more compact image representation. This compact image representation has two key properties. The first property is that the new representation of the image has fewer elements than the original image. For typical images, the number of elements is reduced by one to two orders of magnitude. For many image processing algorithms, this reduction in size makes processing more efficient. The second property of image pyramids is that the objects in the image are given a sense of scale. Small features and noise are not present in the new image representation, which means that for image segmentation algorithms only objects larger than the desired scale are present in the resulting image segmentation.

3.4.1 Previous multi-resolution watershed algorithms

Incorporating an image scale space into watershed image segmentation is not a new idea. Gauch was the first person to propose a multi-scale watershed segmentation algorithm (Gauch, 1993). Gauch's technique uses a Gaussian scale space that is formed by successively filtering the image with a Gaussian filter of increasing variance. After this image scale space has been formed, each watershed region in the original image is assigned a scale based on the level in the scale space where the watershed region in question combines with an adjacent region. A parent-child relationship is also established based on which watershed regions combine in each level of the scale space. Finally, the watershed regions in the original image segmentation are sequentially combined until all of the watershed regions that are below the desired scale have been eliminated.

More recently, Jackway has incorporated a morphological scale space into watershed image segmentation (Jackway, 1996). Jackway's scale space uses morphological operators to produce a hierarchy of watershed regions similar to the parent-child relationship used in Gauch's algorithm. After this hierarchy of watersheds has been established, watershed regions are combined until the desired image segmentation has been achieved.

Gauch and Jackway's algorithms both address the problem of over-segmentation in traditional watershed algorithms, but neither technique reduces the computational expense of the watershed. In fact, both techniques actually *increase* the computational expense of watershed segmentation. The multi-resolution watershed algorithm introduced in this thesis differs from previous techniques since this algorithm uses a morphological image pyramid to calculate the watershed image segmentation. The morphological pyramid not only remedies the problem of over-segmentation, but also reduces the computational expense of watershed analysis.

3.4.2 Morphological image pyramids

For this research, a morphological image pyramid is used to perform watershed image segmentation. The morphological pyramid is constructed by successive filtering of the original image with morphological operators. Morphological operators simplify digital images, preserving shape characteristics and eliminating irrelevant features (Haralick, 1987). Morphological filters also remove noise without introducing a grayscale bias, making them well suited for image segmentation (Sternberg, 1986).

Grayscale morphological operators are based on two fundamental operators, *erosion* and *dilation*. Erosion is defined as a *local-minimum* operation in a digital image. The erosion of a digital image **I** with respect to a structuring element **K** is given by

$$(\mathbf{I} \Theta \mathbf{K})(x) = \max_{y \in \mathbf{K}} \{I(x+y)\}. \qquad (3.7)$$

Dilation is the dual of erosion, and performs a moving local-maximum operation. It follows that the dilation of a digital image **I** with respect to a structuring element **K** is given by

$$(\mathbf{I} \oplus \mathbf{K})(x) = \min_{y \in \mathbf{K}} \{I(x+y)\}. \tag{3.8}$$

Based on these fundamental morphological operators, a number of concatenated operators can be established. Two of these operators are *opening* and *closing*. The opening of a digital image **I** by the structuring element **K** can be described as dilation of the image **I** by the structuring element **K** followed by erosion of the image **I** by the structuring element **K**. It follows that the opening of a digital image **I** with respect to a structuring element **K** is given by

$$\mathbf{I} \circ \mathbf{K} = (\mathbf{I} \ominus \mathbf{K}) \oplus \mathbf{K}. \tag{3.9}$$

The closing of a digital image **I** by the structuring element **K** can be described as erosion of the image **I** by the structuring element **K** followed by dilation of the image **I** by the structuring element **K**. It follows that the closing of a digital image **I** with respect to a structuring element **K** is given by

$$\mathbf{I} \bullet \mathbf{K} = (\mathbf{I} \oplus \mathbf{K}) \ominus \mathbf{K}. \tag{3.10}$$

Morales (1995) has shown that when constructing a morphological pyramid, using an open filter followed by a close filter offers better performance than using an individual open filter or an individual close filter. Using an open-close filter to form the morphological pyramid, pyramid level L is defined by

$$\mathbf{I}_L = \left[\left(\mathbf{I}_{(L-1)} \circ \mathbf{K} \right) \bullet \mathbf{K} \right]_{\downarrow S} \qquad L = 0,1,\ldots,n \tag{3.11}$$

where $\mathbf{I}_0$ is the original image, and $[\cdot]\downarrow_S$ represents a down sampling by a factor of S in each spatial dimension (along rows and columns). The parameter n is largest integer such that for an MxN image $\left\{ M/_{2^n}, N/_{2^n} \right\} \geq 1$. An example of a morphological pyramid formed using an open-close filter is shown in Figure 3.4.

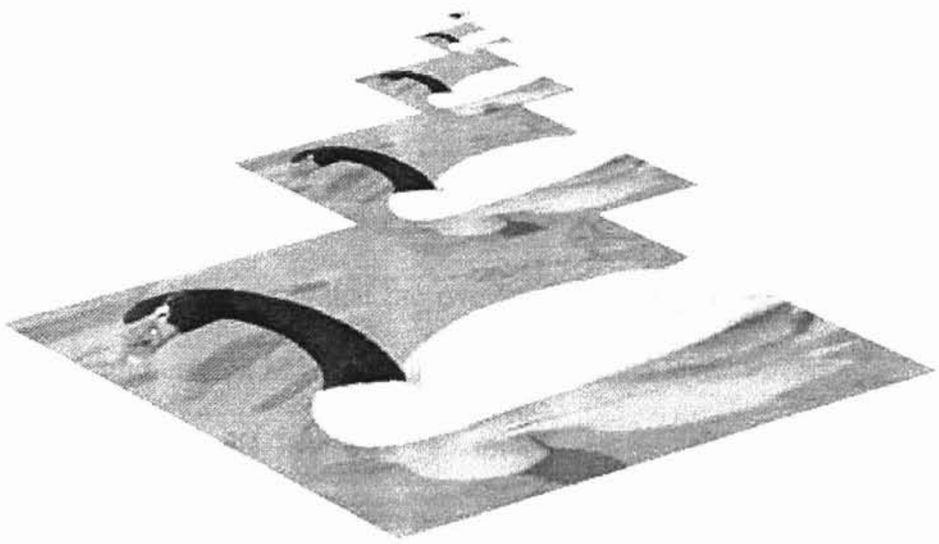

Figure 3.4 Morphological pyramid constructed using an open-close filter. This image pyramid was constructed using a open-close filter with a 3x3 square structuring element and down-sampling by a factor of $S = 2$.

3.4.3 Constructing the watershed pyramid

The watershed pyramid introduced in this section offers a solution to the two most common problems found in most watershed segmentation algorithms. In the watershed pyramid, the watershed algorithm is applied once at a coarse level, and the edges are propagated back to finer representations without performing the watershed algorithm at each level of the pyramid. The image pyramid not only filters out insignificant features and noise, but also reduces the computational expense of watershed image segmentation.

The first step in constructing the watershed pyramid is to build a morphological open-close pyramid using Equation 3.11. Once the morphological pyramid has been constructed, the next step is to choose the root level of the pyramid. The root level must be selected based on two criteria. The first criteria is that the smallest object that is to be preserved in the final segmentation must appear in the root level of the image pyramid. The second criteria is that the two "closest" objects to be preserved in the final segmentation must not be merged in the root pyramid level.

Let a represent the length of the minor axis of the smallest object to be preserved in the root level of the image pyramid. It follows that the pyramid level L_a in which the smallest object will disappear is defined as

$$L_a = \frac{\log_2 a}{\log_2 S} - 1, \tag{3.12}$$

where S is the factor by which the image is down-sampled between pyramid levels in Equation 3.11.

Similarly, let d represent the minimum distance between the two closest objects to be preserved in the root level of the image pyramid. It follows that the pyramid level L_d in which the two objects will be combined is defined as

$$L_d = \frac{\log_2 d}{\log_2 S} - 1. \tag{3.13}$$

It follows that the root level R is determined by

$$R = \min\{L_a, L_d\}. \tag{3.14}$$

Once the root level of the pyramid has been selected, Gauch's watershed algorithm is applies to the root level $\mathbf{I}_R$ of the image pyramid. The resulting watershed segmentation $\mathbf{W}_R$ of the root level is given as

$$\mathbf{W}_R = \mathrm{WS}(\mathbf{G} * \nabla\mathbf{I}_R). \tag{3.15}$$

After the watershed has been applied at the root level R, each in level $\mathbf{W}_{R-1}$ must be linked to a set of elements in level $\mathbf{W}_R$. In the image pyramid, every element in level L has one "parent" in level $L+1$ and four "children" in level $L-1$ since there is a 4 to 1 pixel reduction with each ascending level. This implies that each edge pixel in the root level R corresponds to up to four edge pixels in the level $R-1$. Figure 3.5 illustrates the mapping between levels of an image pyramid.

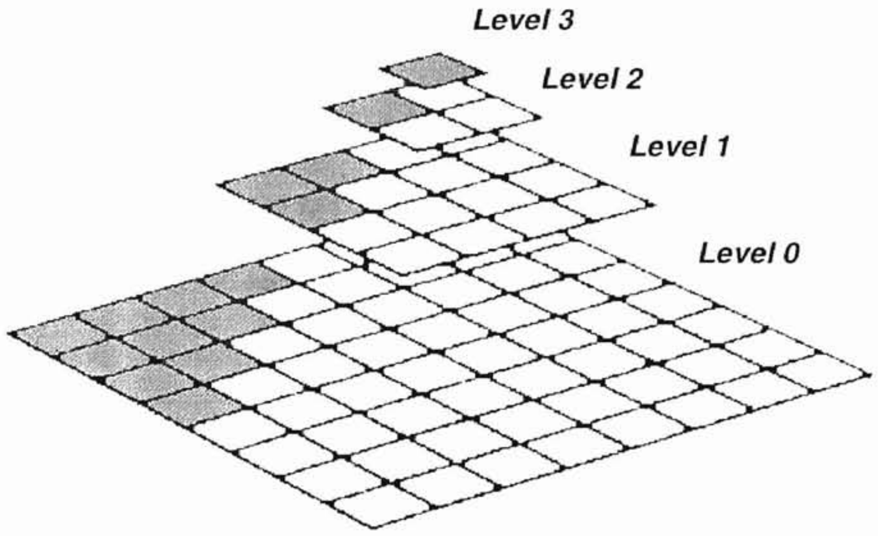

Figure 3.5 Example of the mapping of pixels between levels of an image pyramid.

When linking the levels of the watershed pyramid, we want to maintain connected region boundaries, and insure that no new regions are created as the watershed segmentation is propagated through the image pyramid. In order to accomplish this, the pyramid node linking algorithm is separated into two operations.

The first operation is to link all of the pixels that are not a member of a *watershed boundary* in level *L*-1 to level *L*. Using the definition of a watershed boundary given in Section 3.2, if a pixel *d* in level *L*-1 is a member of catchment basin C_i, and pixel *d* is not a member of a watershed boundary, then each of pixel *d*'s four children in level *L* is a member of the catchment basin C_i. This first step in the linking of pyramid level *L* to level *L*-1 is accomplished by

$$W_{L-1}(d_0) = \begin{cases} W_L(d) & \nabla W_L(d) = 0 \\ -1 & otw. \end{cases} \tag{3.16}$$

for all $d_0 \in C(d, L)$ where $C(d, L)$ represents the children of element *d* at level *L* (Wright, 1997). In Equation 3.16, if $\nabla W_L(d) = 0$, signifying no change in watershed label exists in that neighborhood, the label for the children of $W_L(d)$ is known to be equal to the label of $W_L(d)$. If $\nabla W_L(d) \neq 0$, then the label of the children of element *d* is uncertain. These pixels in level *L*-1 with an uncertain labels are assigned a value of –1 and will be evaluated in the second step of the edge linking algorithm.

The second operation in linking level *L-1* to level *L* is to apply Gauch's watershed algorithm to the elements of $\mathbf{W}_{L-1}$ with uncertain labels. This is accomplished by

$$W_{L-1}(d_0) = \mathrm{WS}\left[B_{L-1}(d_0)\right] \tag{3.17}$$

for all d_0 such that $W_{L-1}(d_0) = -1$, where **B** is defined in Equation 3.1 (Wright, 1997). Since the pyramid structure insures the causality of watersheds, no watershed can appear in level *L-1* that did not exist in level *L*. For this reason an additional step is added to the watershed algorithm. Any new local minimum which is located within a region of uncertainty must be flooded into its nearest neighbor by Equation 3.6, given the condition that any watershed formed in level *L-1* must merge with one and only one watershed that existed in level *L*.

Edge linking continues until *L*-1=0, and finally, edge detection is performed on level $L = 0$ using Equation 3.4. In the next chapter, a number of examples of image segmentation using the watershed pyramid are presented. The resulting image segmentations have significantly fewer edges than traditional watershed segmentation algorithms, and show less sensitivity to noise. The watershed pyramid also performs image segmentation at a much lower computational expense than traditional algorithms.

3.5 Chapter Summary

Watershed segmentation is a powerful image processing tool, although traditional techniques tend to over-segment images and have a high computational expense. In this chapter, a multi-resolution watershed image segmentation algorithm was described. This algorithm is based on a morphological image pyramid, and uses Gauch's minimum following algorithm to calculate the watershed segmentation.

The watershed pyramid has a number of properties which make it well suited for computer vision applications. The first property is that the algorithm is computationally efficient, offering a reduction in the computational expense by one to two orders of magnitude over traditional algorithms. Another important property of image segmentation using the watershed pyramid is that the segmentation is tunable based on the selection of the root pyramid level. By selecting a "high" level as the root level of the image pyramid, small features can be eliminated producing an image segmentation based on the major features in the original image. By "lowering" the root level, smaller features can be added to the image segmentation. A third property of the watershed pyramid is that the resulting edge map is guaranteed to have connected edges that are of single pixel width. This property is particularly useful when the purpose of image segmentation is object identification or pattern recognition.

In summary, the watershed pyramid offers an alternative means to calculate the watershed image segmentation. The watershed pyramid algorithm reduces the computational expense and addresses the problem of over-segmentation found in traditional algorithms while maintaining an image segmentation that features closed contours and region boundaries that closely correspond to the edges of object in the original image.

In the next chapter, results of watershed pyramid image segmentation are presented. These results confirm that the watershed pyramid is an effective method of calculating the watershed image segmentation, and also confirm that the watershed pyramid helps to solve the problems of over-segmentation and computational expense associated with traditional algorithms.

CHAPTER IV

WATERSHED PYRAMID RESULTS

4.1 Chapter Overview

In this chapter, results are presented from several applications of the watershed pyramid. These results demonstrate that incorporating a morphological pyramid into watershed image segmentation pyramid helps to remedy the two major problems associated with traditional watershed segmentation algorithms: over-segmentation and computational expense.

The first problem with most traditional algorithms is that watershed analysis is very sensitive to small features. This sensitivity typically results in an over-segmentation of the image. By incorporating an image pyramid into the watershed, the objects in the resulting image segmentation are given a sense of scale. With careful selection of the root pyramid level, small features and noise can be eliminated from the final image segmentation. Section 4.2 presents some sample image segmentations calculated using the watershed pyramid. These examples show the image segmentation for each level of the watershed pyramid, demonstrating the refinement of the edge map as the region information is propagated through the image pyramid. Section 4.2 also compares image segmentations calculated using the watershed pyramid with image segmentations calculated using a traditional watershed algorithm. Finally, Section 4.2 compares the performance of the watershed pyramid with the performance of a traditional watershed segmentation algorithm for images that have been corrupted with noise.

The second problem with most traditional algorithms is the computational expense associated with calculating the watershed segmentation. This problem is also addressed by the use of an image pyramid. With a watershed pyramid, the watershed image segmentation is only performed once at a coarse scale representation of the original image. The resulting image segmentation is then propagated through the image pyramid, until it reaches the original image. Section 4.3 presents a detailed analysis comparing the computational expense of performing watershed image segmentation using a morphological pyramid versus using a traditional watershed algorithm. This analysis shows that even with the computational expense of building the image pyramid and propagating the edge information through the pyramid, the watershed pyramid is more efficient than the two most common traditional watershed algorithms. Section 4.4 concludes the chapter with a brief summary of the features of the watershed pyramid.

4.2 Visual watershed pyramid results

4.2.1 Watershed pyramid results

This section presents image segmentation results from applying the watershed pyramid to sample images. The results demonstrate that the watershed pyramid is applicable to a variety of different imaging scenarios, and helps remedy the problem of over-segmentation. Figures 4.1 through 4.12 show four examples of applying the watershed pyramid to digital images. These examples show the image pyramid used to calculate the watershed, and the resulting image segmentation for each level of the image pyramid.

In each example, an open-close filter with a 3x3 square kernel is used to form the morphological pyramid, and the image is down-sampled by a factor of 2 along each row and each column between levels of the pyramid. The gradient magnitude of each level of the pyramid was blurred with a Gaussian filter with $\sigma = 1$ to help insure local uniqueness for each pixel in the image. The root level of the pyramid was selected based on the size of the smallest feature desired in the final segmentation.

Figure 4.1 shows the image pyramid used to segment the "Swan" image. This image pyramid was formed using a 3x3 morphological open-close filter. Figure 4.2 shows the image segmentation for each level of the watershed pyramid. When using the watershed pyramid, image segmentation is only performed one time at the root level of the image pyramid. For this example the size of the original image is 256x256. The root level is defined as $L = 3$, which corresponds to a size of 32x32. After the initial segmentation, region information is linked back through the pyramid to the original image. Figure 4.3 shows the image segmentation after incorporating a simple region merging algorithm into the watershed pyramid. The region merging algorithm used for this case is described in Chapter 3, and uses the mean pixel value of each region in the initial image segmentation. If the mean pixel value of two adjacent regions in the watershed image segmentation differs by less than a threshold T, the regions are combined. The region merging algorithm is applied once at the root level, and the "simplified" image segmentation is propagated through the pyramid to the original image. It should be noted that the region merging can also be performed at the base level of the image pyramid ($L = 0$). Performing region merging at the base pyramid level typically offers a more accurate image segmentation, but does not take advantage of the computational savings offered by the image pyramid.

Figures 4.4 through 4.6 show the same image segmentation results for the "Peppers" test image. Next, Figures 4.7 through 4.9 show the results for the "Old Central" test image. Finally, Figures 4.10 through 4.12 show the results for the "Meg Ryan" test image.

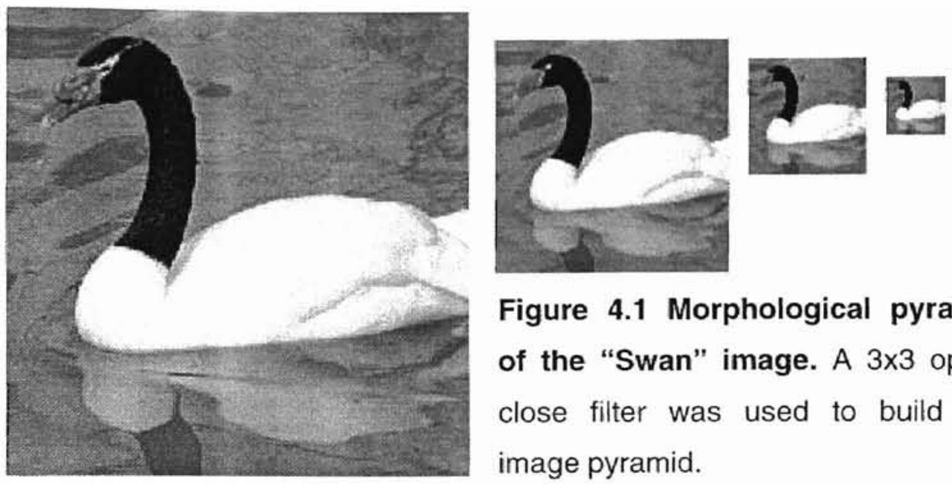

Figure 4.1 Morphological pyramid of the "Swan" image. A 3x3 open-close filter was used to build the image pyramid.

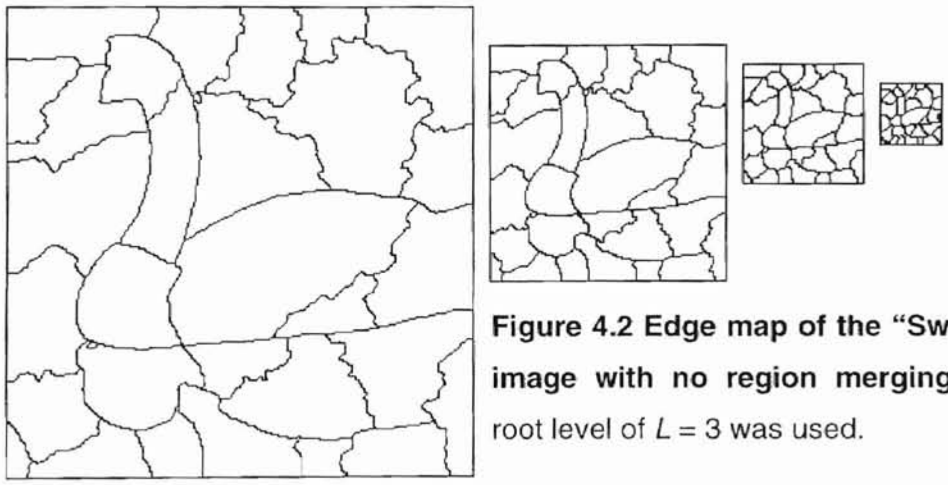

Figure 4.2 Edge map of the "Swan" image with no region merging. A root level of $L = 3$ was used.

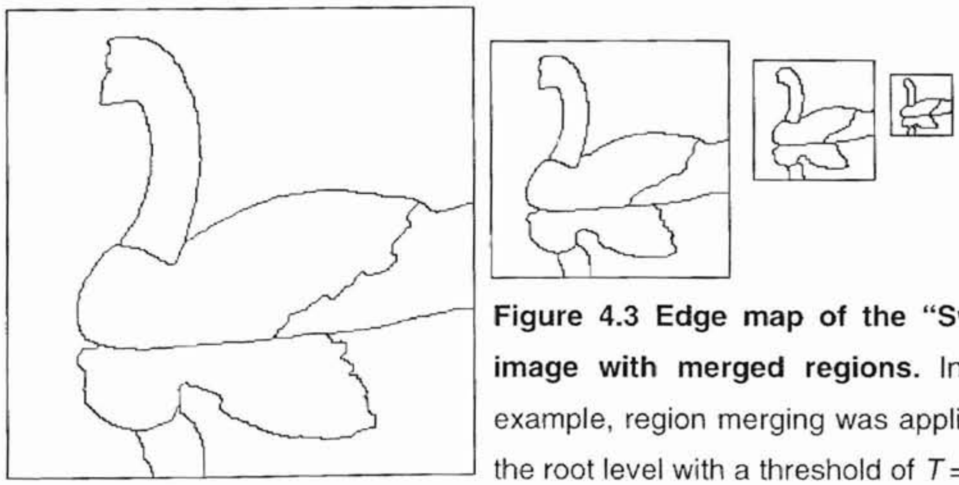

Figure 4.3 Edge map of the "Swan" image with merged regions. In this example, region merging was applied at the root level with a threshold of $T = 15$.

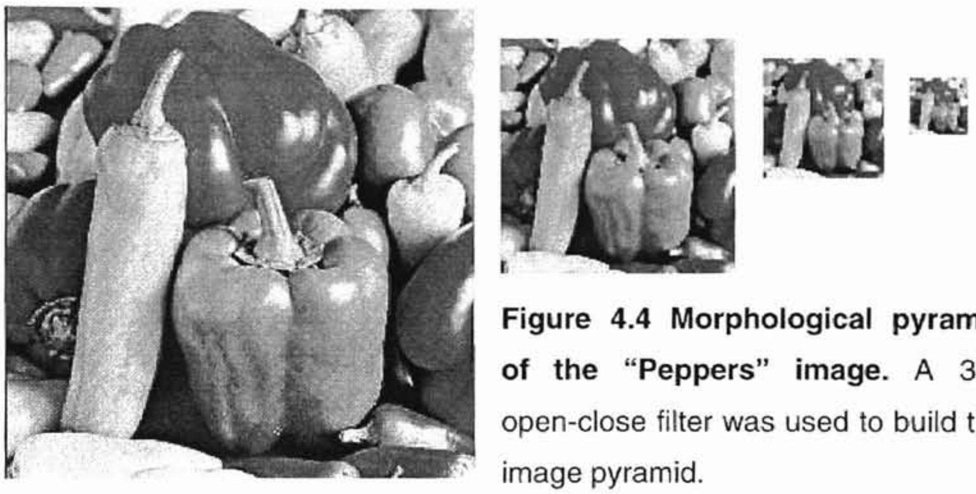

Figure 4.4 Morphological pyramid of the "Peppers" image. A 3x3 open-close filter was used to build the image pyramid.

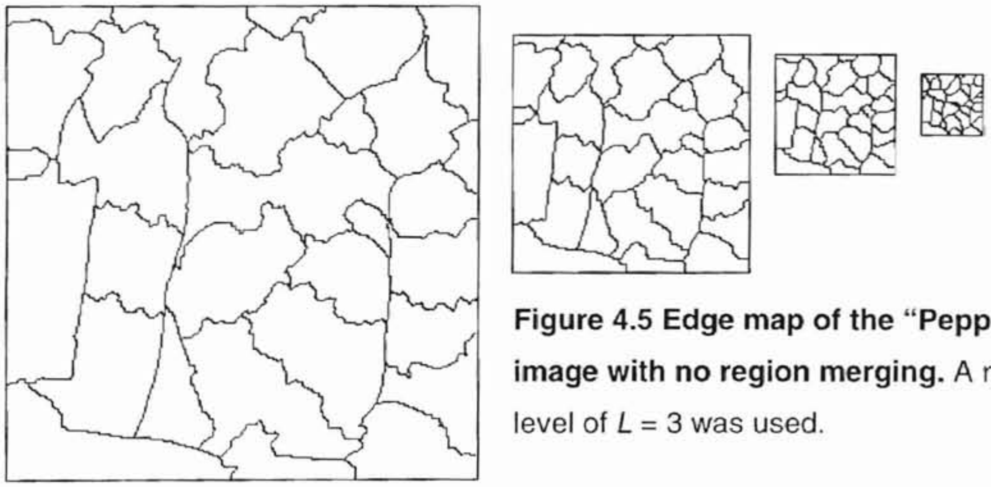

Figure 4.5 Edge map of the "Peppers" image with no region merging. A root level of $L = 3$ was used.

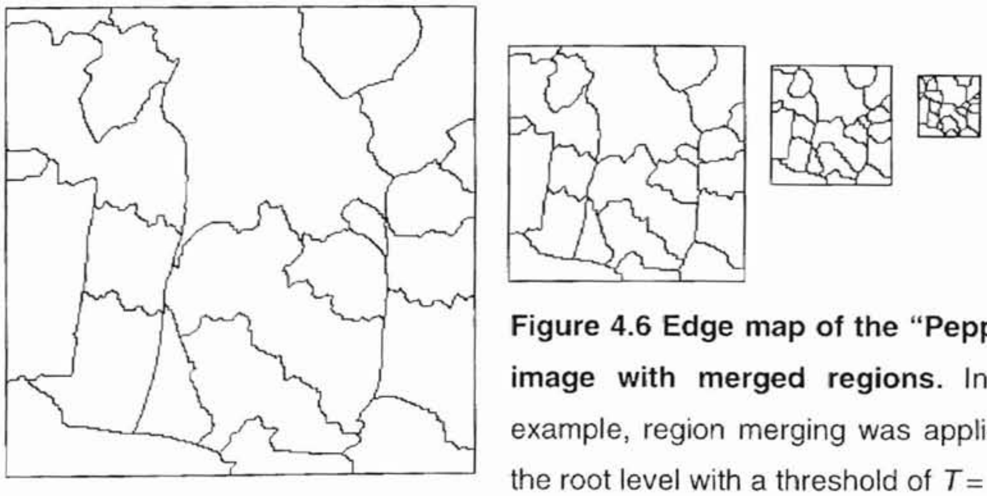

Figure 4.6 Edge map of the "Peppers" image with merged regions. In this example, region merging was applied at the root level with a threshold of $T = 7$.

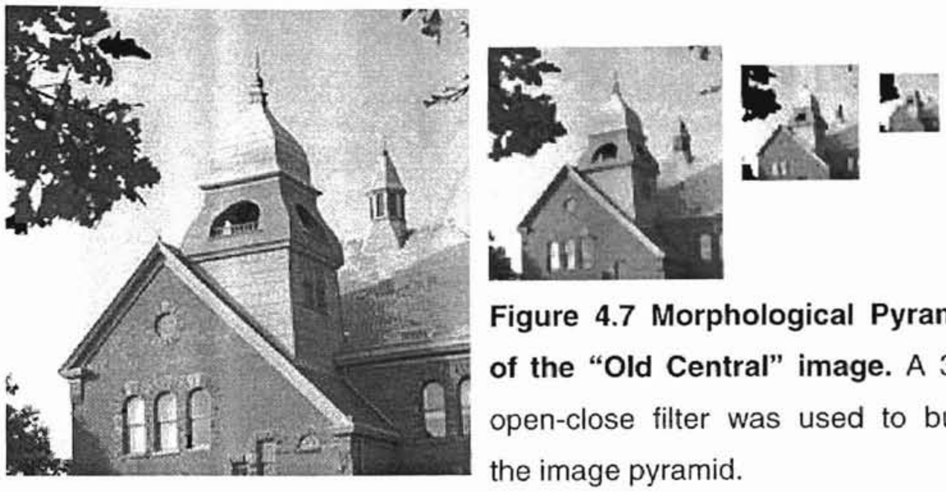
Figure 4.7 Morphological Pyramid of the "Old Central" image. A 3x3 open-close filter was used to build the image pyramid.

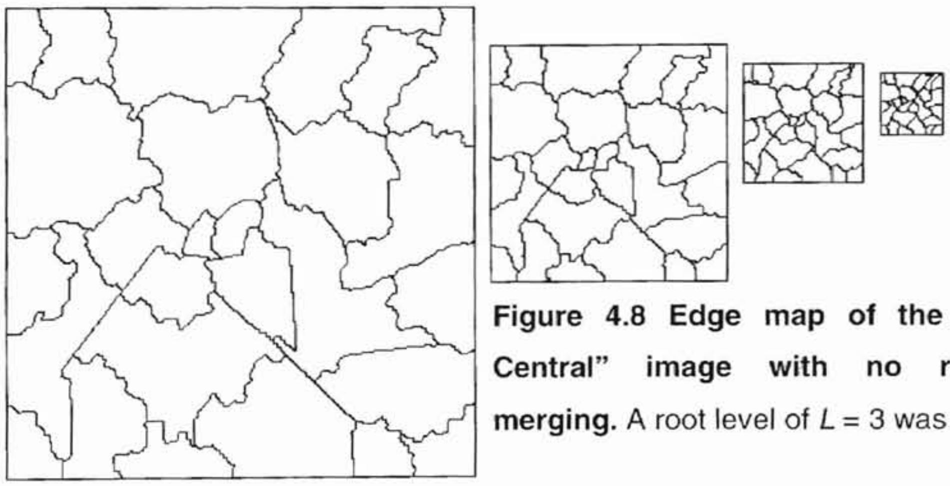
Figure 4.8 Edge map of the "Old Central" image with no region merging. A root level of $L = 3$ was used.

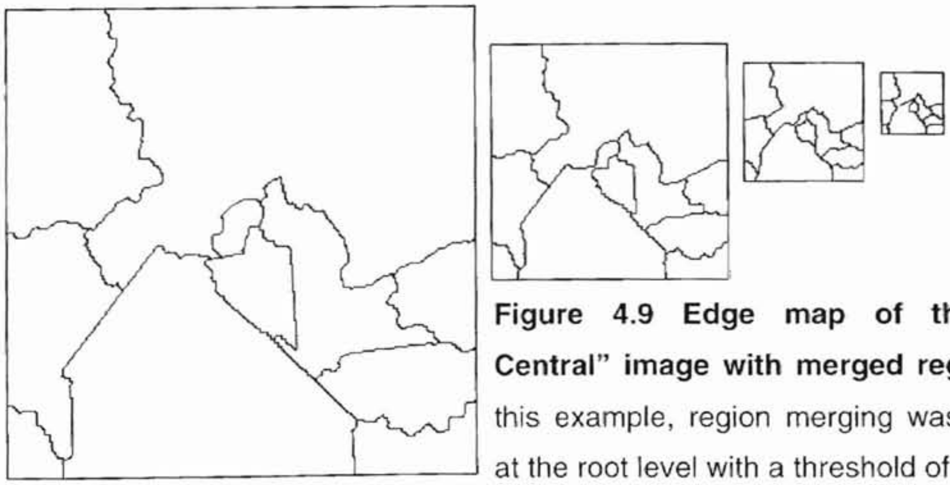
Figure 4.9 Edge map of the "Old Central" image with merged regions. In this example, region merging was applied at the root level with a threshold of $T = 12$.

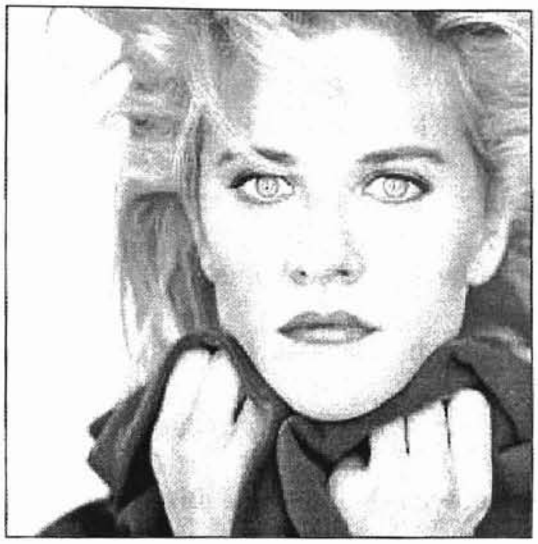
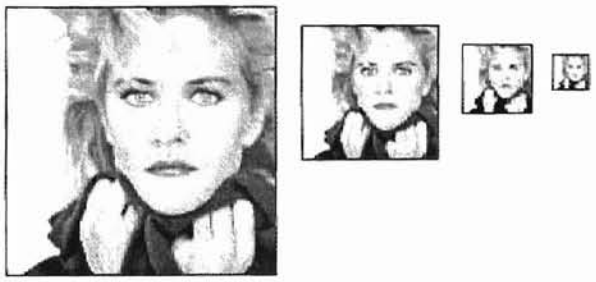

Figure 4.10 Morphological Pyramid of the "Meg Ryan" image. A 3x3 open-close filter was used to build the image pyramid.

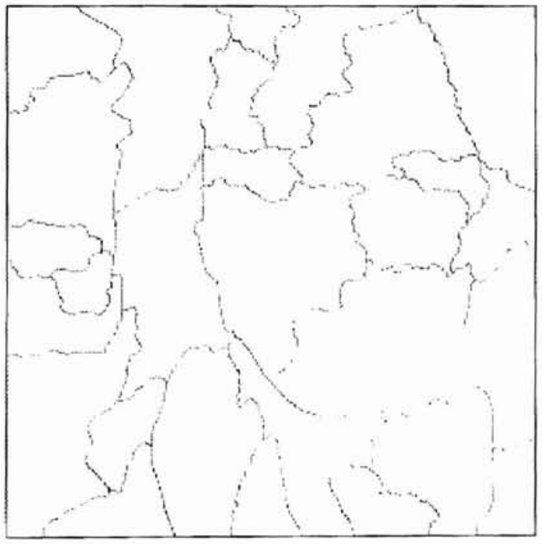
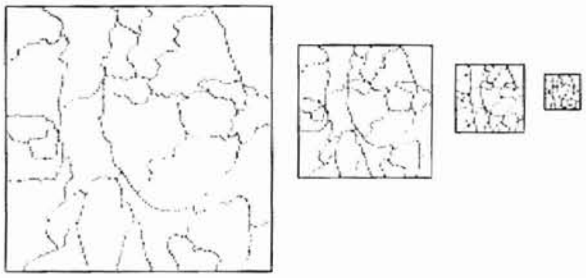

Figure 4.11 Edge map of the "Meg Ryan" image with no region merging. A root level of $L = 4$ was used.

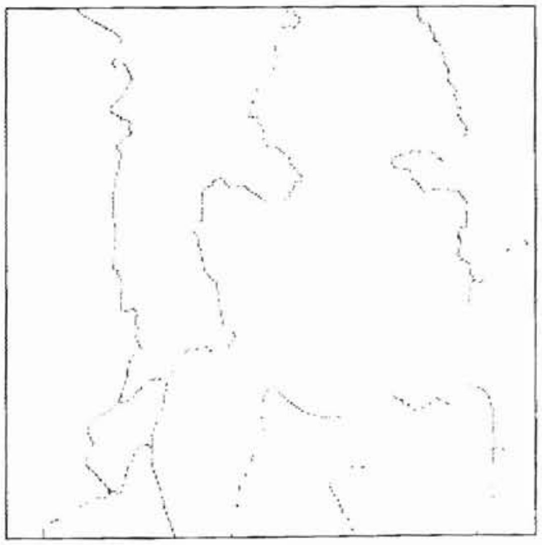
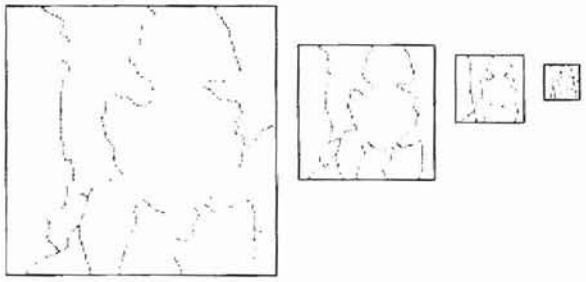

Figure 4.12 Edge map of the "Meg Ryan" image with merged regions. In this example, region merging was applied at the root level with a threshold of $T = 22$.

4.2.2 *Watershed pyramid vs. traditional watershed analysis*

Figures 4.13 through 4.16 compare watershed segmentation on a single resolution image with the results of image segmentation using a watershed pyramid. These results demonstrate that without using any region merging techniques, the watershed pyramid addresses the problem of over-segmentation. For each of these cases, the single resolution image segmentation is applied to the original image after prefiltering with a Gaussian filter with $\sigma = 3$ (this corresponds to a 7x7 square kernel.) No region merging was performed on the resulting image segmentation. For each of these test images, the watershed pyramid image segmentation results are identical to those presented in the Figures 4.1 through 4.12. In each case, the watershed pyramid was formed using a a 3x3 open-close morphological filter. A Gaussian filter with $\sigma = 1$ was then applied to the gradient magnitude of the root level of the image pyramid level. Finally, watershed image segmentation was performed at the root level, and the information was propagated back to the original image. Region merging was not incorporated into the watershed pyramid segmentation.

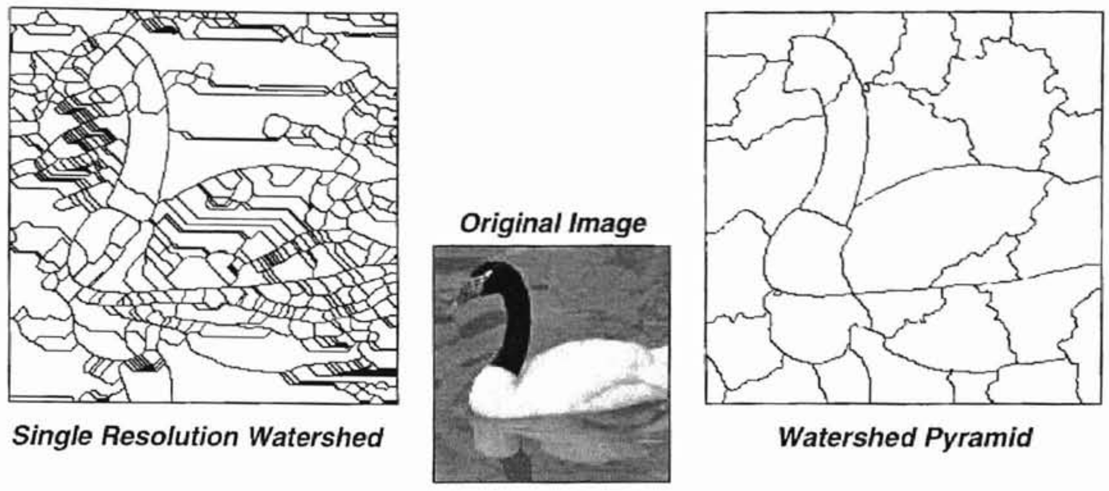

Figure 4.13 Segmentation of the "Swan" image with a watershed pyramid vs. traditional watershed segmentation. The segmentation on the left was calculated without using a watershed pyramid or region merging. The image was pre-filtered with a Gaussian filter with $\sigma = 3$. The segmentation on the right was calculated using a 3x3 open-close image pyramid with root level of $L = 3$.

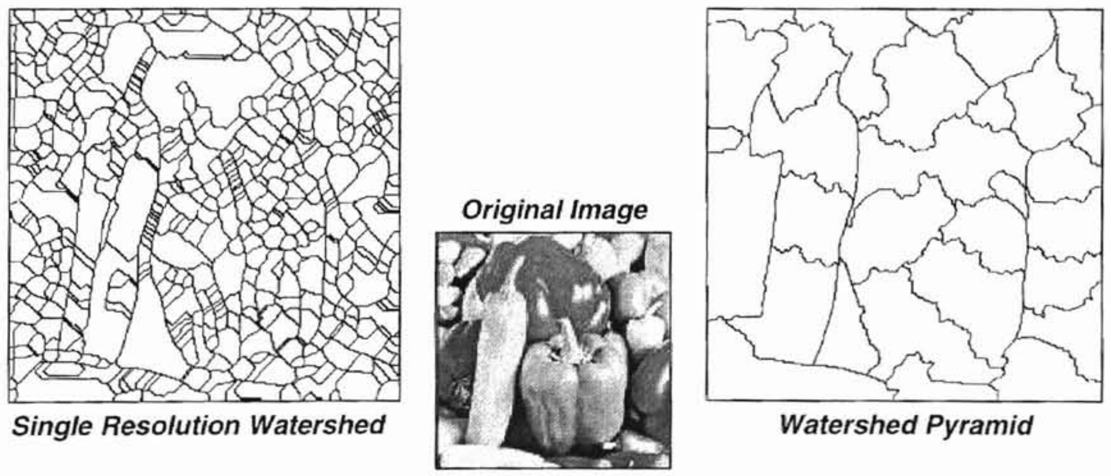

Figure 4.14 Segmentation of the "Peppers" image with a watershed pyramid vs. traditional watershed segmentation. The segmentation on the left was calculated without using a watershed pyramid or region merging. The image was pre-filtered with a Gaussian filter with $\sigma = 3$. The segmentation on the right was calculated using a 3x3 open-close image pyramid with root level of $L = 3$.

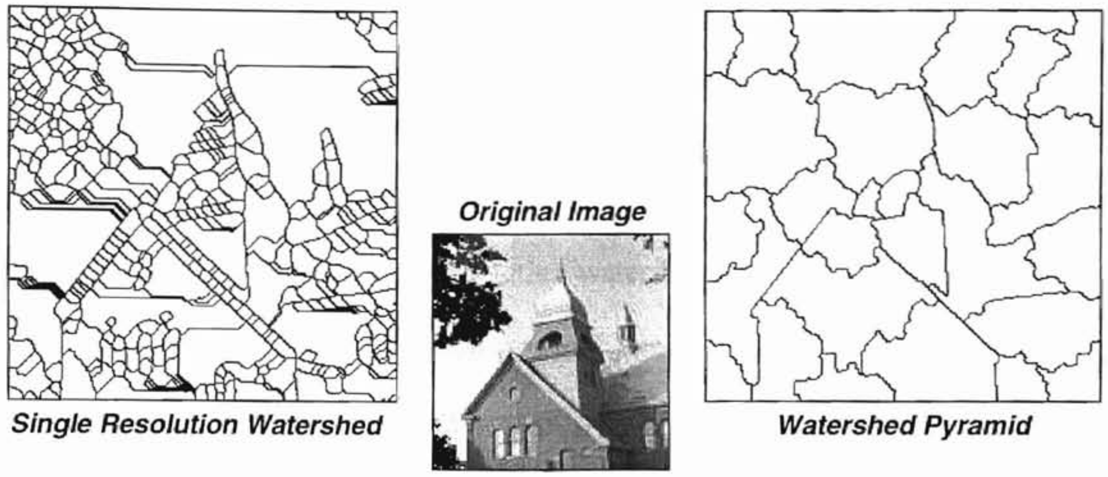

Figure 4.15 Segmentation of the "Old Central" image with a watershed pyramid vs. traditional watershed segmentation. The segmentation on the left was calculated without using a watershed pyramid or region merging. The image was pre-filtered with a Gaussian filter with $\sigma = 3$. The segmentation on the right was calculated using a 3x3 open-close image pyramid with root level of $L= 3$.

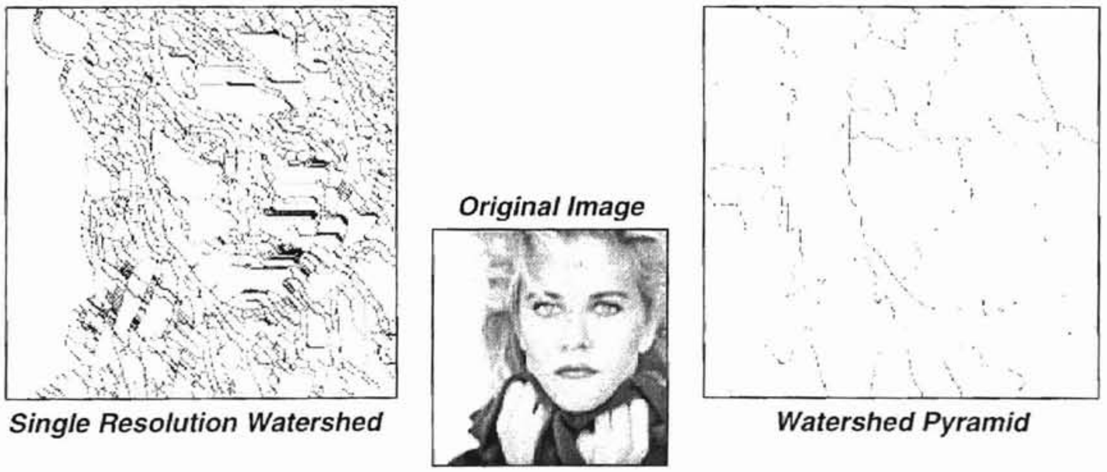

Figure 4.16 Segmentation of the "Meg Ryan" image with a watershed pyramid vs. traditional watershed segmentation. The segmentation on the left was calculated without using a watershed pyramid or region merging. The image was pre-filtered with a Gaussian filter with $\sigma = 3$. The segmentation on the right was calculated using a 3x3 open-close image pyramid with root level of $L = 4$.

4.2.3 Performance of the watershed pyramid on noisy images

Figures 4.17 and 4.18 show the performance of the watershed pyramid in the presence of noise. These results demonstrate that the watershed pyramid performs well on noisy images, while the single resolution watershed algorithm suffers from severe over-segmentation. The morphological pyramid acts to filter out the noise, minimizing its effect on the watershed segmentation of the root pyramid level. The presence of noise does have minor effects on the edge localization as the boundary information is propagated through the image pyramid, but the resulting image segmentation maintains regions that directly correspond to large scale features in the original image.

For the examples shown in Figures 4.17 and 4.18, the original images are corrupted with zero mean Gaussian noise with $\sigma^2 = 40$. The single resolution image segmentation is applied to the noise corrupted image after prefiltering with a Gaussian filter with $\sigma = 3$. No region merging was performed on the resulting image segmentation. For the watershed pyramid segmentation, the image pyramid was formed using a a 3x3 open-close morphological filter. For the "Swan" example, a level of $L = 3$ was selected as the root level of the image pyramid. For the "Meg Ryan" example, a level of $L = 4$ was selected as the root level of the image pyramid. Next, a Gaussian filter with $\sigma = 1$ was applied to the gradient magnitude of the root level of the image pyramid. Finally, watershed image segmentation was performed at the root level, and the information was propagated back to the original image. No region merging was performed on the resulting image segmentation for either example.

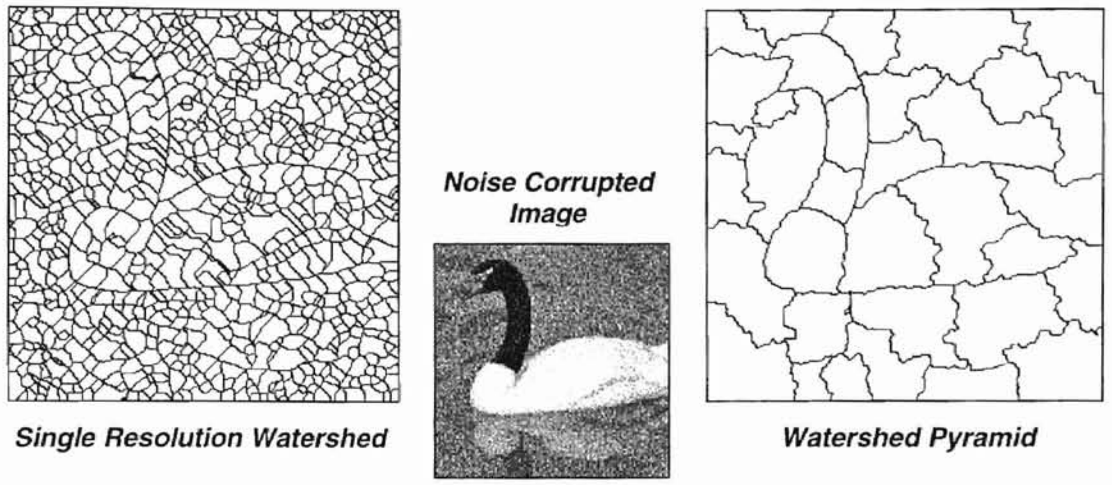

Figure 4.17 Performance of the watershed pyramid on a noise corrupted image. The "Swan" image was corrupted with zero mean Gaussian noise with $\sigma^2 = 40$. The segmentation on the left was calculated without using a watershed pyramid or region merging. The noise corrupted image was pre-filtered with a Gaussian filter with $\sigma = 3$. The segmentation on the right was calculated using a 3x3 open-close image pyramid with root level of $L = 3$.

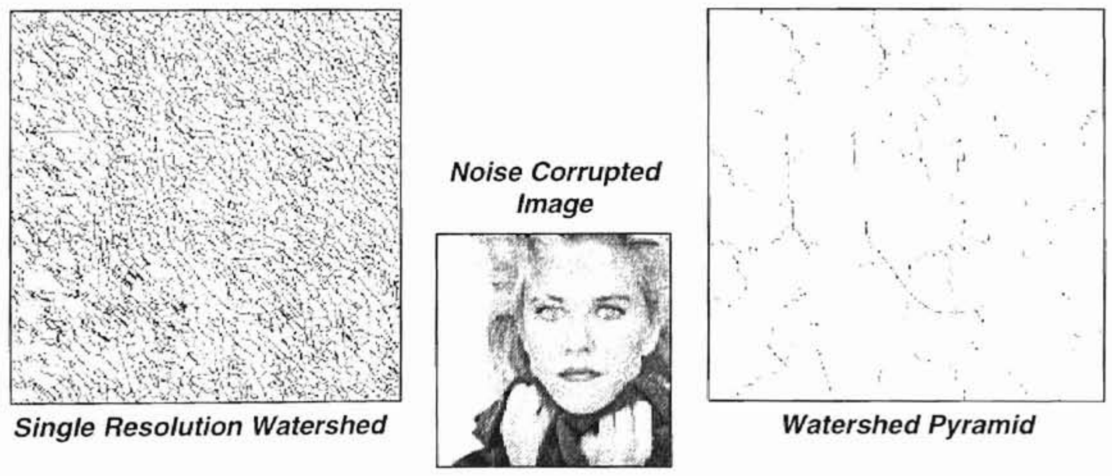

Figure 4.18 Performance of the watershed pyramid on a noise corrupted image. The "Meg Ryan" image was corrupted with zero mean Gaussian noise with $\sigma = 40$. The segmentation on the left was calculated without using a watershed pyramid or region merging. The noise corrupted image was pre-filtered with a Gaussian filter with $\sigma = 3$. The segmentation on the right was calculated using a 3x3 open-close image pyramid with root level of $L = 4$.

4.3 Computational expense of the watershed pyramid

In practice the multiresolution algorithm decreases the computational complexity of traditional watershed segmentation algorithms by an order of magnitude. In this section, the computational expense of implementing the watershed pyramid is compared with the computational expense of traditional watershed segmentation algorithms. The traditional watershed algorithms used for comparison are the steepest descent algorithm presented by Gauch and Pizer, and the queue based algorithm presented by Vincent and Soille (Gauch, 1993; Vincent, 1991). It should be noted that the computational expense of the queue based algorithm presented by Meyer and Beucher is nearly identical to the computational expense of Vincent's watershed algorithm (Meyer, 1992; Dobrin, 1994).

4.3.1 Computational expense of Gauch's watershed algorithm

Making the assumption that a comparison between elements is equivalent to an addition operation, the cost of performing the full resolution watershed on an NxN image using Gauch's steepest descent watershed algorithm can be estimated. The first step is to perform the ∇ operation which requires $3N^2$ adds and $2N^2$ multiplies. Next, the gradient magnitude of the image is blurred with a Gaussian filter to ensure local uniqueness of each pixel, requiring $8(G{\cdot}N)^2$ adds and $9(G{\cdot}N)^2$ multiplies for convolution with a GxG Gaussian, where $G = 2\sigma + 1$. The final step is to perform the WS operation, which requires $8N^2$ adds for the arrowing operation and $2N^2$ adds for the marking operation. In total, Gauch's watershed algorithm requires $(13+9G^2)N^2$

addition operations and $(2+9G^2)N^2$ multiplication operations, not taking into consideration the computational cost of pre-filtering or region merging.

4.3.2 *Computational expense of Vincent's watershed algorithm*

Again making the assumption that a comparison between elements is equivalent to an addition operation, the cost of performing the full resolution watershed on an NxN image using Vincent's queuing watershed algorithm can be estimated. The first step is to perform the ∇ operation which requires $3N^2$ adds and $2N^2$ multiplies. Next, the elements of the image must be sorted based on gray-level intensity which requires N^2 adds. Finally, the elements at each gray level are analyzed, which requires $8N^2$ adds. In total, Vincent's watershed algorithm requires $13N^2$ addition operations and $2N^2$ multiplication operations, not taking into consideration the computational cost of pre-filtering or region merging.

4.3.3 *Computational expense of the watershed pyramid*

The computation expense of applying the watershed pyramid to an NxN image can be estimated using the assumption that a comparison between elements is equivalent to an addition operation. For the watershed pyramid algorithm, the watershed is applied at a level R that is of size MxM, where $M = N/2^R$, then linked to the finer levels of the pyramid. The cost of constructing the pyramid using an open-close filter with a kernel of size KxK is $4K^2 \sum_{L=0}^{R-1} \left(N/2^L\right)^2$ adds. After the morphological pyramid has been constructed, Gauch's watershed algorithm is applied to the root level of the image pyramid. Using the computational expense derived in Section 4.3.1, this requires $(13+9G^2)M^2$ addition operations and $(2+9G^2)M^2$ multiplication operations where G represents the size of the Gaussian kernel used to blur the gradient magnitude of the image. The final step is to perform pyramid level linking to propagate the watershed boundary information through the image pyramid. Assuming there are E_R elements in level R which represent watershed boundaries, the watershed must be performed on $4E_R$ elements to link level R to level $R\text{-}1$. This linking will produce $E_{R-1} \approx 2E_R$ elements in level $R\text{-}1$ since connectivity is maintained. The resulting computational cost of linking the multiresolution watershed is found to be approximately $\left(13+9G^2\right)\cdot 4\sum_{L=0}^{R-1} 2^L E_R$ adds. In total, the watershed pyramid algorithm requires approximately $\left(13+9G^2\right)\cdot\left[M^2+4\sum_{L=0}^{R-1} 2^L E_R\right]+4K^2\sum_{L=0}^{R-1}\left(N/2^L\right)^2$ addition operations, and $(2+9G^2)M^2$ multiplication operations, not taking into consideration the computational expense of pre-filtering or region merging. Since these formulas are a bit confusing, it is best to look at the computational expense of actual image segmentations to draw a comparison between the watershed pyramid and traditional watershed segmentation techniques.

4.3.4 Computational expense in practical applications

In this section, the computational expense of performing image segmentation using the watershed pyramid for four different test images is compared with the computational expense of performing image segmentation using Vincent's watershed algorithm and Gauch's watershed algorithm. The results show that in practice, the watershed pyramid reduces the computational complexity of watershed image segmentation by an order of magnitude.

The first test image is the "Swan" image shown in Figures 4.1 through 4.3. For this case, the original image has a size of 256x256, and the image pyramid is constructed using an open-close filter with a 3x3 kernel. A level of $L = 3$ is selected as the root level, making the computational expense of building the image pyramid up to the root level equal to $3.1\text{x}10^6$ adds. The size of the root level is 32x32, making the computational expense of applying Gauch's watershed algorithm at the root level equal to $41\text{x}10^3$ adds and $30\text{x}10^3$ multiplies. Finally, the level linking operation requires $764\text{x}10^3$ adds and $573\text{x}10^3$ multiplies. This makes the total computational expense of segmenting the "Swan" image using a watershed pyramid equal to $3.9\text{x}10^6$ addition operations and $603\text{x}10^3$ multiplication operations.

Segmenting the "Swan" test image using Gauch's watershed algorithm requires $25.7\text{x}10^6$ adds and $28.9\text{x}10^6$ multiplies to pre-filter with a 7x7 Gaussian filter, followed by $2.6\text{x}10^6$ adds and $2.0\text{x}10^6$ multiplies to calculate the watershed. This makes the total computational expense of the Gauch's watershed algorithm equal to $28.3\text{x}10^6$ addition operations and $30.9\text{x}10^6$ multiplication operations.

Finally, segmenting the "Swan" test image using Vincent's watershed algorithm requires $25.7\text{x}10^6$ adds and $28.9\text{x}10^6$ multiplies to pre-filter with a 7x7 Gaussian filter, followed by $852\text{x}10^3$ adds and $131\text{x}10^3$ multiplies to calculate the watershed. This makes the total computational expense of the Vincent's watershed algorithm equal to $26.6\text{x}10^6$ addition operations and $30.2\text{x}10^6$ multiplication operations.

The second test image is the "Meg Ryan" image shown in Figures 4.10 through 4.12. For this case, the original image has a size of 512x512, and the image pyramid is constructed using an open-close filter with a 3x3 kernel. A level of $L = 4$ is selected as the root level, making the computational expense of building the image pyramid up to the root level equal to $12.6\text{x}10^6$ adds. The size of the root level is 32x32, making the computational expense of applying Gauch's watershed algorithm at the root level equal to $41\text{x}10^3$ adds and $30\text{x}10^3$ multiplies. Finally, the level linking operation requires $1.9\text{x}10^6$ adds and $1.4\text{x}10^6$ multiplies. This makes the total computational expense of segmenting the "Meg Ryan" image using a watershed pyramid equal to $14.5\text{x}10^6$ addition operations and $1.4\text{x}10^6$ multiplication operations.

Segmenting the "Meg Ryan" test image using Gauch's watershed algorithm requires $102.8\text{x}10^6$ adds and $115.6\text{x}10^6$ multiplies to pre-filter with a 7x7 Gaussian filter, followed by $10.5\text{x}10^6$ adds and $7.9\text{x}10^6$ multiplies to calculate the watershed. This makes the total computational expense of the Gauch's watershed algorithm equal to $113.3\text{x}10^6$ addition operations and $123.5\text{x}10^6$ multiplication operations.

Finally, segmenting the "Meg Ryan" test image using Vincent's watershed algorithm requires $102.8\text{x}10^6$ adds and $115.6\text{x}10^6$ multiplies to pre-filter with a 7x7 Gaussian filter, followed by $3.4\text{x}10^6$ adds and $524\text{x}10^3$ multiplies to calculate the watershed. This makes the total computational expense of the Vincent's watershed algorithm equal to $106.2\text{x}10^6$ addition operations and $116.1\text{x}10^6$ multiplication operations.

The computational expense of all four test images used in Section 4.2 is summarized in Table 4.1. These results show that in every case, the watershed pyramid reduces the number of addition operations necessary to perform watershed image segmentation by approximately 1/7th, and reduces the number of multiplication operations by approximately 1/50th. In total, the watershed pyramid reduces the total number of computations by well over an order of magnitude.

	Watershed pyramid		***Gauch's watershed***		***Vincent's watershed***	
	Adds $\times10^6$	Multiplies $\times10^6$	Adds $\times10^6$	Multiplies $\times10^6$	Adds $\times10^6$	Multiplies $\times10^6$
"Swan" image *size 256x256*	3.9	0.6	28.3	30.9	26.6	30.2
"Meg Ryan" image *size 512x512*	14.5	1.4	113.3	123.5	106.2	116.1
"Old Central" image *size 256x256*	3.9	0.6	28.3	30.9	26.6	30.2
"Peppers" image *size 256x256*	4.0	0.7	28.3	30.9	26.6	30.2

Table 4.1 - Computational expense of the watershed pyramid vs. traditional watershed algorithms. This table shows the approximate number of calculations necessary to perform watershed image segmentation using three different methods: the watershed pyramid, Gauch's watershed algorithm, and Vincent's watershed algorithm. These calculations include a Gaussian pre-filter for the single resolution results, and do not take into account any region merging operations.

4.4 Chapter Summary

This chapter has presented several results from image segmentation using the watershed pyramid. The visual results presented in Sections 4.2 through 4.4 offer a qualitative view of the performance of the watershed pyramid. Figures 4.1 through 4.12 demonstrate that the pyramid level linking algorithm presented in Chapter 3 is effective in propagating the region information from the root level of the pyramid to the original image. These examples also demonstrate that by incorporating a region merging algorithm into the watershed pyramid it is possible to further remedy the problem of over-segmentation. Figures 4.13 through 4.16 demonstrate that the watershed pyramid offers an effective solution to the problem of over-segmentation associated with most traditional watershed segmentation algorithms. Finally, Figures 4.17 and 4.18 demonstrate that while the presence of noise in the input image adds to the over-segmentation problem of the traditional watershed algorithm, it has little effect on the image segmentation calculated using the watershed pyramid.

Section 4.3 presents a quantitative analysis of the computational expense of watershed image segmentation using the watershed pyramid. It also includes analysis of the computational expense of watershed image segmentation using Gauch's watershed algorithm and Vincent's watershed algorithm. This analysis shows that the watershed pyramid reduces the computational complexity of watershed segmentation by more than an order of magnitude for most cases.

CHAPTER V

CONCLUSION

5.1 Summary of the Watershed Pyramid

Image segmentation is a key element in many computer vision algorithms, creating a demand for robust algorithms to segment digital images. Although there is a wide variety of techniques to segment images into logical regions, there is no one algorithm that works well for every application. The focus of this thesis is an improved watershed segmentation algorithm. Watershed segmentation is a powerful image processing tool, although traditional techniques tend to over-segment images and have a high computational expense.

The watershed image segmentation algorithm presented in this thesis offers an alternative solution to the traditional watershed segmentation algorithms developed to date. This new technique uses a pyramid based watershed algorithm which incorporates a sense of scale into the image segmentation, guarantees closed contours, and is computationally efficient. This multi-resolution watershed image segmentation algorithm presented in Chapter 3 is based on a morphological image pyramid, and uses Gauch's minimum following algorithm to calculate the watershed segmentation.

The watershed pyramid has a number of properties which make it well suited for computer vision applications. The first property is that the algorithm is computationally efficient, offering a reduction in the computational expense by one to two orders of magnitude over traditional algorithms. Another key property of the watershed pyramid is that the resulting edge map is guaranteed to have closed contours that are of single pixel width. This property is particularly useful when the purpose of image segmentation is object identification or pattern recognition. A third important property of the watershed pyramid is that the segmentation is tunable based on the selection of the root pyramid level. By selecting a "high" level as the root level of the image pyramid, small features can be eliminated producing an image segmentation based only on the major features in the original image. By "lowering" the root level, smaller features can be added to the image segmentation.

In Chapter 4, results of watershed pyramid image segmentation are presented. These results confirm that the watershed pyramid is an effective method of calculating the watershed image segmentation, and also demonstrate that the watershed pyramid helps to solve the problems of over-segmentation and computational expense associated with traditional algorithms. The visual results presented in Chapter 4 offer a qualitative view of the performance of the watershed pyramid. These results verify that the watershed pyramid remedies the problem of over-segmentation found in traditional watershed segmentation algorithms, and also demonstrate that while the presence of noise in the input image adds to the over-segmentation problem of traditional watershed algorithms, it has little effect on image segmentations calculated using the watershed pyramid. Chapter 4 also presents a quantitative analysis of the computational expense of watershed image segmentation using the watershed pyramid. This analysis shows that the watershed pyramid reduces the computational complexity of watershed segmentation by more than an order of magnitude over traditional algorithms for most cases.

In summary, the watershed pyramid offers an alternative method to calculate the watershed image segmentation. The watershed pyramid algorithm reduces the computational expense and addresses the problem of over-segmentation found in traditional algorithms, while maintaining an image segmentation that features closed contours and region boundaries that closely correspond to the edges of object in the original image.

5.2 Future Work

The image segmentation algorithm presented in this research offers an alternative method to segment digital images. Although this algorithm does out-perform other watershed segmentation algorithms in most cases, there is still work to be done surrounding the watershed pyramid algorithm.

One key area of research that is not explored in this thesis is the development of a more robust region merging algorithm to be incorporated into the watershed pyramid. The region merging technique used for the examples in this thesis was a simple algorithm based solely on the mean pixel intensity of the watershed regions, and does not perform well for certain types of images. Koshimizu (1998) has presented a more robust algorithm to merge watershed regions, but the subject of watershed region merging remains an open area of research.

A second area of research that was unexplored in this thesis was the performance of the watershed pyramid algorithm using different types of filters to form the image pyramid. The algorithm presented in this thesis is not constrained to the morphological pyramid, and can be extended to other pyramid image structures such as the Gaussian pyramid or the anisotropic diffusion pyramid. Although the morphological pyramid performs well for the examples presented in this research, other types of image pyramids may be better suited for certain applications.

REFERENCES

Acton, S., A.C. Bovick, and M.M. Crawford (1994), "Anisotropic Diffusion Pyramids for Image Segmentation," *Proceedings of the IEEE International Conference on Image Processing*, Nov. 1994.

Beucher, S. and C. Lantuejoul (1979), "Use of Watersheds in Contour Detection", *Proceedings of the International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, Sept. 17-21, 1979.

Beucher, S. (1982), "Watersheds of Functions and Picture Segmentation", *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Paris, France.

Brice, C. and C. Fennema (1970), "Scene Analysis Using Regions", *Artificial Intelligence*, vol. 1, pp. 205-226.

Burt, P.J. (1988), "Smart sensing within a pyramid vision machine," *Proceedings of the IEEE*, vol.76, no.8, pp.1006-15, 1988.

Castleman, K.R. (1996), *Digital Image Processing*, Prentice Hall, Inc., New Jersey.

Collins, S.H. (1975), "Terrain Parameters Directly from a Digital Terrain Model", *Canadian Surveyor*, vol. 29, no. 5, pp. 227-248.

Dobrin, B.P., T. Viero, M. Gabbouj (1994), "Fast Watershed Algorithms: Analysis and Extensions", *Nonlinear Image Processing V*, SPIE vol. 2180, pp. 209-220.

Eichman, G. C. Lu, J. Zhu, and Y. Li (1988), "Pyramidal Image Processing Using Morphology", *Applications of Digital Image Processing XI*, SPIE vol. 974, pp. 30-37.

Gauch, J.M. and S.M. Pizer (1993), "Multiresolution Analysis of Ridges and Valleys in Grey-Scale Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6.

Gonzalez, R.C. and R.E. Woods (1993), *Digital Image Processing*, Addison Wesley Publishing Co, New York.

Haralick, R.M and L.G. Shapiro (1985), "Survey: Image Segmentation Techniques", *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 100-132.

Haralick, R.M., S.R. Sternberg, and X. Zhuang (1987), "Image Analysis Using Mathematical Morphology", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 532-550.

Haralick, R.M. and L.G. Shapiro (1992), *Computer and Robot Vision*,Vol.1, Addison-Wesley Publishing Co., New York.

Horowitz, S.L. and T. Pavlidis (1974), "Picture Segmentation by a Directed Split and Merge Procedure", *Proceedings of the 2nd International Joint Conference on Pattern Recognition*, pp. 424-433.

Jackway, P.T. (1996), "Gradient Watersheds in Morphological Scale Spaces", IEEE Transactions on Image Processing, vol. 5, no. 6, pp. 913-921.

Koshimizu, T. (1998), "Watershed Segmentation and Region Merging with Application to Remote Sensing", *Oklahoma State University Masters of Science Thesis*.

Kasturi, R. and R.C. Fain (1991), *Computer Vision: Principles*, IEEE Computer Society Press, Washington D.C.

Klinger, K. (1973), "Data Structures and Pattern Recognition", *Proceedings of the First International Joint Conference on Pattern Recognition*, Washington, D.C., pp. 497-498.

Levine, M.D. and S.I. Shaheen (1981), "A Modular Computer Vision System for Image Segmentation and Interpretation", *IEEE Transactions on Pattern Analysis and Machine Analysis*, vol. PAMI-3, pp. 287-300.

Marks, D., J. Dozier, and J. Frew (1984), "Automated Basin Delineation from Digital Elevation Data", *Geoprocessing*, vol. 2, pp. 299-311.

Marr, D. and E. Hildreth (1980), "Theory of Edge Detection", *Proceedings of the Royal Society of London*, Series B, vol. 207, pp. 187-217.

Meyer, F. and Beucher, S. (1990), "Morphological Segmentation", *Journal of Visual Communication and Image Representation*, vol.1, no.1, Sept. 1990, pp. 21-46.

Meyer, F. and Beucher, S. (1992). "The Morphological Approach to Segmentation: The Watershed Transformation", *Mathematical Morphology in Image Processing*. Marcel Dekker, Inc., New York, pp. 433-481.

Morales, A., R. Acharya and S.J. Ko (1995), "Morphological Pyramids with Alternating Sequential Filters", *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp.965-977.

Nagy G. and J. Tolaba (1972), "Nonsupervised Crop Classification through Airborne Multispectral Observations", *IBM Journal of Research and Development*, vol. 16, pp. 138-153.

Prewitt, J. (1970), "Object Enhancement and Extraction", in *Picture Processing and Psychopictorics*, edited by B. Limpkin and A Rosenfield, Academic Press, New York, pp. 1674-75.

Robertson, T.V. (1973), "Extraction and Classification of Objects in Multispectral Images", *Machine Processing of Remotely Sensed Data*, IEEE 73 CHO 837-2GE, Purdue University, West Lafayette, IN, pp. 3B-27-3B-34.

Russ, J.C. (1995), *The Image Processing Handbook, Second Edition*, CRC Press, Florida.

Saarinen, K. (1994), "Color Image Segmentation by a Watershed Algorithm and Region Adjacency Graph Processing", *Proceedings of the IEEE International Conference on Image Processing*, Nov. 1994.

Sobel, I.E. (1970), "Camera Models and Machine Perception", *AIM-2*, Stanford Artificial Intelligence Lab, Palo Alto.

Sternberg, S.R. (1986), "Grayscale Morphology", *Computer Vision, Graphics, and Image Processing*, vol. 35, pp. 333-335.

Vincent, L. and E.R. Dougherty (1994), "Morphological Segmentation for Textures and Particles", in *Digital Image Processing Methods*, edited by E.R. Dougherty, Marcel Dekker Publishing, Inc., New York, pp. 43-102.

Vincent, L. and P. Soille (1991), "Watersheds in Digital Spaces: An Efficient Algorithm based on Immersion Simulations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, June 1991, pp. 583-598.

Witkin, A.P. (1983), "Scale-space filtering", *Proceedings of the International Joint Conference on Artificial Intelligence*, pp.1019-22.

Wright, A.S., and S. Acton (1997), "Watershed Pyramids for Edge Detection", *Proceedings of the IEEE International Conference on Image Processing*, Oct. 1997.

VITA

Anthony S. Wright

Candidate for the Degree of

Master of Science

Thesis: IMAGE SEGMENTATION USING WATERSHED PYRAMIDS

Major Field: Electrical Engineering

Biographical:

Education: Received Bachelor of Science degree in Electrical Engineering from Oklahoma State University, Stillwater, Oklahoma in May 1996. Completed requirements for the Master of Science degree at Oklahoma State University in December, 1998.

Professional Experience: Graduate Research Assistant, Oklahoma Imaging Laboratory, School of Electrical and Computer Engineering, Oklahoma State University, May 1996 to December, 1997. Engineer for WorldCom, Tulsa, Oklahoma, January, 1998 to May, 1998. Engineer for Williams Communications, Tulsa, Oklahoma, May, 1998 to present.