2007

Effects of pre-processing and postprocessing on the watershed transform

Padmavati Tanniru
San Jose State University

EFFECTS OF PRE-PROCESSING AND POSTPROCESSING ON THE WATERSHED TRANSFORM

A Thesis

Presented to

The Faculty of the Department of Mathematics

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Padmavati Tanniru

May 2007

APPROVED FOR THE DEPARTMENT OF MATHEMATICS

R. Dodd

Dr. Roger Dodd

Leslie Foster

Dr. Leslie Foster

[illegible]

Dr. Hedley Morris

APPROVED FOR THE UNIVERSITY

[illegible] 04/24/07

ABSTRACT

EFFECTS OF PRE-PROCESSING AND POSTPROCESSING ON THE WATERSHED TRANSFORM

by Padmavati Tanniru

This work addresses the topic of the Watershed Transform as a tool for image segmentation. It examines the possibility of extracting the maximum information from an image by generating various resolutions of the image using various preprocessing techniques, and then applying the Watershed Transform on the set of images thus obtained. We compare these results of the Watershed Transform with each other, as well as with the segmentation achieved through multiscale edge detection, a purely local technique. We also examine the role of post-processing techniques in improving the results.

The basic idea behind the Watershed Transform in the case of gray scale images is to create a landscape by assigning heights according to the gray levels (0 - 255 in a standard gray scale image) associated with its pixels. Such a procedure translates the given image into a landscape from which the boundaries are almost evident. When the concept of multiresolution is used in conjunction with it, the Watershed Transform becomes a powerful technique, because of the fact that at low resolutions the coarse structure of the image is detected whereas the fine structure is revealed at high resolutions. The Watershed Transform can pick up structures which are present in a band of resolutions. Performing a coarse to fine analysis and detecting watersheds at each level enables a consensus to be formed about the true boundaries in the image.

ACKNOWLEDGEMENT

I dedicate my work to my family, parents, and all my teachers who, formally or informally, have always helped me bring out the best in me.

This work would not have been possible without the support and encouragement of Dr. Roger Dodd, my thesis advisor, under whose supervision I chose to work on this topic and took it to completion. I thank him for always being there to guide me. I want to thank the committee members, Dr. H. Morris and Dr. L. Foster, for their evaluations and suggestions.

I thank the faculty at SJSU of which some of my best teachers include Dr. Dodd, Dr. S. Obaid, Dr. E. Obaid, Dr. B. Jackson, Dr. L. Foster, Dr. Wasin So, Dr. J. Mitchem, Dr. R. Pfiefer, Dr. L. Valdes, and Dr. B. Lee. There are other faculty members whose timely advise/assistance helped simplify a lot of matters and constraints. They include Dr. R. Kubelka, Dr. Schmeichel, Dr. J. Day, and Ms. Susan McClory. Thank you for always being available to answer my numerous, related or unrelated, questions.

I want to thank my parents, my brother, and my sister who always believed in me, and provided all the support, especially in my trying times. There were instances which almost forced me to give-up, but if one has such a support system, then one can survive and meet the goal.

I want to thank my husband, Deepak, who in his own ways motivated me to take my task to completion. My two beautiful children, Umesh and Juhi, have also contributed in helping me keep the intent of finishing my work alive. It was for their sake and love that

I felt I will be able to provide better once I complete my work.

I want to thank wildlife@exis.net for granting me the permission to use the Zebra image at www.nature-wildlife.com/zeb56.html, William Zittrich for granting permission to use the Monarch Butterfly image at http://www.geocities.com/wyllz/id236.htm, and Bruce Montagne for granting me the permission to use the Ferns image at http://www.whisperwood.net/pixnpgs/p/p16.html for the purpose of experimentation and research. Lastly, I want to thank all those who made available the source code at http://ippr.csie.ntu.edu.tw:8080/demoDownload.html which helped me configure the varied aspects of the whole idea that is the Watershed Transform.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this thesis is to investigate the use of the Watershed Transform applied to images at different resolutions as a segmentation tool. For the sake of simplicity, only gray-scale images will be considered for analysis. The goal of an image analysis is determined by the expectations one has from the images for which the techniques are to be applied. For example, an accurate analysis of the MRI scans of a brain become critical when done for the detection of tumors. Surgeons are interested in determining the free margin [27] surrounding the tumor so that a complete excision can be made. A detailed study of an image at different resolutions can aid such an analysis, and applied with the Watershed Transform may lead to a clearer understanding of the location and extent of the tumor. This is of the utmost importance when health-related decisions are to be made.

For the purpose of this thesis the images taken up for processing are the following three images:

1. Zebra Up Close from www.nature-wildlife.com/zeb56.html

2. The Monarch Butterfly from http://www.geocities.com/wyllz/id236.htm

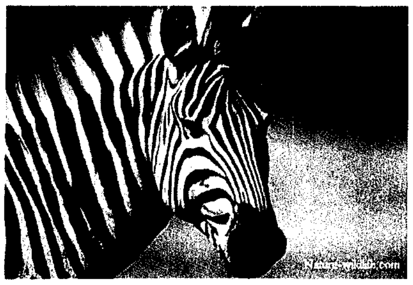


Figure 1.1: The Zebra

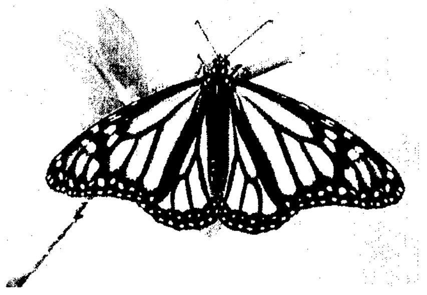


Figure 1.2: The Monarch Butterfly

3. Wood Lily in Ferns from http://www.whisperwood.net/pixnpgs/p/p16.html

The type of analysis adopted by segmentation techniques can be categorized as below:

- Local analysis as seen in edge detectors like the sobel, roberts, prewitt, LOG operator
- Global but still local in strategy as in the Canny operator
- Global analysis as in the Watershed Transform

Edge detection makes use of filters which enables detection of the variations in pixel properties, and this information is then used to link-up pixels that belong to an edge. The important property of the Watershed Transform, as we discuss in Chapter 3, is its nonlocal nature. One implementation of the algorithm requires that at each step the total

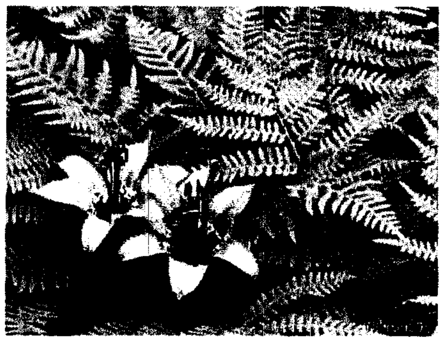


Figure 1.3: Wood Lily in Ferns

image is processed. As part of this thesis we study boundary detection using watersheds, and compare the watersheds obtained from an image by first preprocessing it using different techniques. Segmentation techniques using filters are the focus of discussion in Chapter 2 wherein their role in edge detection is analyzed. In particular, this chapter contains a section on the Canny edge detector. The Watershed Transform is discussed in Chapter 3.

Chapter 4 analyzes the effect of preprocessing and postprocessing an image before and after detecting the watersheds using the Watershed Transform combined with the morphological techniques. Chapter 5 involves experimentation with non-morphological techniques and examines the watersheds, thus, obtained. It summarizes the results and contrasts them with the use of different edge detecting procedures, in particular, the Canny edge detector.

1.1 Edge Detection

Edges are an integral part of the contours that define the segmentation of an image. They are associated with sharp changes in the grayscale values which occur in the neighborhood of an edge. Finite difference methods can be used to construct derivatives which are the basis of edge detection. Two methods which use this approach are based on:

- the gradient(first derivative)
- the Laplacian(second derivative)

These properties are discussed in Chapter 2. The detection of edges using them is implemented with the help of certain operators called filters that can execute computations at the pixel level. This chapter also contains a more general discussion on filters as they pertain to image segmentation.

1.1.1 Some Special Edge Detection Operators

These are taken up in detail in Chapter 2.

1. Sobel Operator
2. Canny Operator
3. Roberts Operator
4. Laplacian of the Gaussian (LOG) Operator
5. Zero-Crossings Operator

1.2 Watershed Algorithm

The easiest way to define a watershed is to use a geographical analogy. If a grayscale image is viewed in such a manner that high and low intensity pixels correspond to high and low

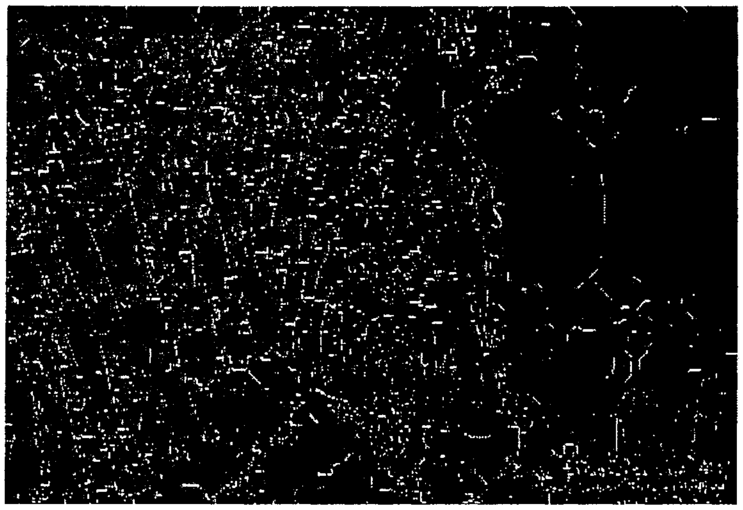


Figure 1.4: Edges in the Zebra Image Detected using the Canny Edge Detector

ground then the image becomes a 3D landscape. A local minimum in the pixel intensity is then a dip in the ground surrounded by higher land. This acts as a catchment basin when water is falling on the landscape. The watersheds are the set of curves that separate the catchment basins which form as flooding occurs.

We consider two ways of obtaining the watersheds in an image; namely, the rainfalling analogy, and the immersion/flooding analogy.

The *rainfalling* analogy can be best understood by following a raindrop along paths of steepest decent and detecting the number of catchment basins the point where the raindrop fell belongs to. Those pixels where a raindrop has more than one path of steepest decent and hence can flow into more than one basin are the pixels belonging to the watersheds in the image. In this analogy, one needs to explore only the immediate neighbors of the pixel hit by the raindrop to locate the next pixel in its path of steepest decent. Hence, at any level or altitude one needs to compute and preserve information only about the neighborhood pixels. But what should be the strategy when a raindrop comes across a plateau? Plateaus generate thick watershed lines in the watershed image. Various techniques have been developed to thin them.

The *immersion* analogy can be best understood by piercing holes at the local minima and then immersing the whole landscape in a lake or a body of water. Since water starts filling up the catchment basins from these minima, there will be a stage when water from one catchment basin starts mixing with that from another. As and when this is detected, dams are built to avoid the waters from different basins meeting. The height of the dams is continually increased as the process continues until the highest point in the landscape is reached. In this manner a network of dams in the landscape is obtained which constitute the watershed curves. In this analogy, one needs to process and preserve the information about all the pixels at one level before recursively proceeding to the next level. Thus, in the end, we obtain ordered sets of watershed pixels, such that each set bounds a catchment basin. Thus, the number of such sets indicates the number of catchment basins in the landscape, and their boundaries are the required watersheds.

Implementation of the above analogies:

An implementation of either approach takes considerable computer time even on moderately sized images. This has led researchers to try to parallelise them in order to speed up the process. It is difficult to see how this can be done efficiently with the immersion approach [1, 50]. The rain falling analogy seems to offer the best chance of an efficient parallel algorithm [28, 62].

History of the Watershed Transformation:

The Watershed Transformation was introduced as a morphological tool by H. Digabel and Ch. Lantujoul. Lantujoul and Beucher used it for the segmentation of grayscale images, and later Luc Vincent and Pierre Soille devised fast algorithms for both sequential and parallel computation of the watersheds of an image. All of these approaches used the immersion analogy.

Watersheds have been extensively studied from theoretical and algorithmic points of view. The Watershed Transform when combined with other morphological tools becomes an extremely powerful segmentation procedure [50, 2, 49]. The Figure 1.5 shows the watersheds for the original zebra image found by applying the transform directly on the image without any preprocessing using Mathlab 7.0 with a 4-connectivity.

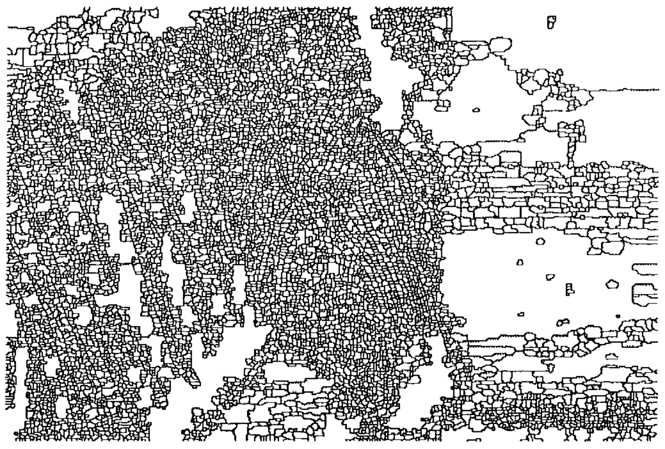


Figure 1.5: Immersion Watersheds of the Zebra Image

This software uses the immersion analogy in identifying the watersheds. The watersheds in the butterfly image were similarly determined and can be seen in Figure 1.6.

The watersheds in the zebra are closer to the real contours than those in the butterfly. This can be because of the presence of oversegmentation in the butterfly image.

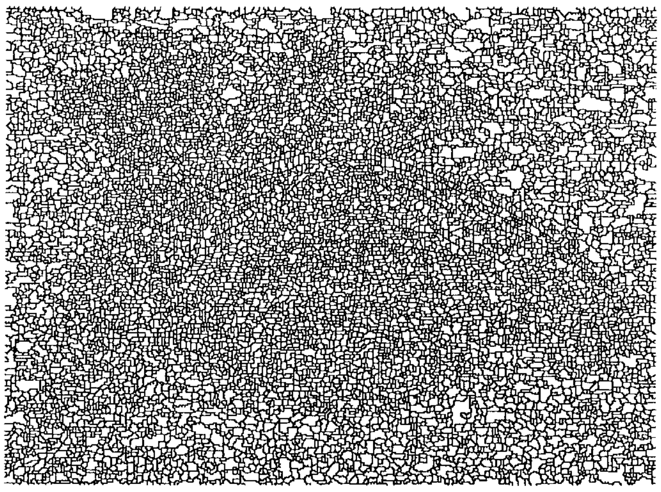


Figure 1.6: Immersion Watersheds of the Butterfly Image

Preprocessing the image is a good way of reducing oversegmentation and preserving true edges. This idea is investigated in Chapter 3.

1.3 Image Multiresolution

An image is a combination of fine and coarse details. The fine details are best viewed at high resolutions and the coarse details at low resolutions [61]. The multiresolution of an image consists of a collection of derived images formed from analyzing the image across a wide range of scales.

An image pyramid is obtained by averaging over pixels or omitting pixels. This idea was first investigated by Burke [30]. The averaged image has fewer pixels and so is smaller than the image from which it was obtained. The collection of averaged images can therefore be arranged so that the apex of the pyramid has the lowest resolution and the base has the highest resolution approximation. For example, this can be done by performing recursive subdivisions into quadrants, giving rise to trees of degree 4, called quadtrees whose leaves represent homogenous blocks of image. If these blocks are visualized as pixels then the pyramid data structure is obtained. A better way of doing this is to use wavelets which do not reduce the size of the image.

After performing a multiresolution analysis of an image, the Watershed Transform can be applied to each of the images in the set so obtained. Details appear and disappear at different resolutions. An analysis of the images in the set can then allow a more complete detection of all the features in the image. Deriving the watersheds of the resolved image set is crucial in this analysis. Because of the time constraint, this would be the motivation for a future research.

1.4 The Right Approach to Image Analysis

It is very difficult to automate image segmentation as the required segmentation depends on what the user wants and in almost all instances human intervention is unavoidable. Knowledge about the interaction of the Watershed Transform, multiresolution, filters and their limitations is crucial prior to performing an analysis of an image. In order to get the best possible segmentation results, one needs to find the best ordering to integrate, and blend these techniques. For example, Laurent Najman, and Michel Schmitt [5] provide a procedure based on geodesic reconstruction that effectively reduces the oversegmentation in an image. Chapter 5 investigates the effect on the analysis of changing the order of a sequence of the techniques.

Research on noise identification and its subsequent processing cannot be considered complete as a consensus on the best way to reduce noise still has to be arrived at; see for example [1, 33, 4, 8, 32, 31, 3].

1.5 Summary

This thesis studies the multiresolution of an image and the effect of using the Watershed Transform on the component images so obtained. A common problem encountered in applying the Watershed Transform to an image is over-segmentation. Over-segmentation can be reduced by preprocessing the image with a suitable filter, but can never be completely removed in this way. The combined watershed and multiresolution approach, coupled with filtering provides the best chance of minimizing over-segmentation, and improving the watersheds.

The thesis presents a study of the Watershed Transform by:

- Investigating its theoretical and practical bases
- Comparing it with the Edge Detection procedures
- Attempting to improve its existing algorithm
- Exploring ways to reduce its sensitivity to noise
- Studying the watersheds obtained after the image is treated with a variety of pre- and post processing techniques

Chapter 2

Filters

2.1 Introduction

Images are made up of pixels that are identified by their location and attributes like their gray values. Differences in the intensities of the pixels gives rise to discontinuities in an image which can correspond to isolated points or edges. On the other hand, there are also regions in which the pixels have similar intensities. Since the purpose of segmenting an image is to extract such attributes of interest, one needs techniques that can detect these two different types of behavior.

Regions of an image in which the pixel intensities vary slowly or rapidly can be detected by operators called *filters*. Traditionally filters have been implemented as *linear time or space invariant operators*, though recently filtering by nonlinear operators has become an active research area [55]. In this thesis intervals are over $\mathbb{Z}$, $[a, b] \subset \mathbb{Z}$. For simplicity let $f : \mathbb{Z}^2 \to I$ define an image, where I denotes the range of pixel values (for example $I = [0, 255]$). Images have compact support, but they can be padded out by defining the intensity of pixels outside the support to be zero. Let $g : \mathbb{Z}^2 \to I$ be an image which is obtained from f by a linear operator L,

$$g = Lf.$$

The operator L is *space invariant* if a translation of the image $t_{\mathbf{a}}f(x,y) := f(x - a_1, y - a_2)$, $\mathbf{a} = (a_1, a_2) \in \mathbb{Z}^2$, causes a corresponding translation of g

$$t_{\mathbf{a}}g = Lt_{\mathbf{a}}f.$$

Linear operators of this type are defined by their action on the *delta impulse function*, a function $\delta : \mathbb{Z}^2 \to \mathbb{Z}$ such that $\delta(x,y) = \delta_{x,y}$ where $\delta_{x,y}$ is the Kronecker delta ($\delta_{x,y} = 1$ for $x = y$ and 0 elsewhere). We note that $\delta(x-u, y-v) = t_{(u,v)}\delta(x,y)$ so that

$$f(x,y) = \sum_{p,q\in\mathbb{Z} f(p,q) t_{(p,q)}\delta(x,y) and L f(x,y) = \sum_{p,q\in\mathbb{Z}} f(p,q) L t_{(p,q)}\delta(x,y).}$$

Define the impulse response of L to be

$$h(x,y) := L\delta(x,y).$$

Then the translational invariance requires that

$$t_{(u,v)}h(x,y) = Lt_{(u,v)}\delta(x,y)$$

and so

$$Lf(x,y) = \sum_{p,q\in\mathbb{Z}} f(p,q)h(x-p, y-q) = \sum_{p,q\in\mathbb{Z}} h(p,q)f(x-p, y-q).$$

The convolution of f by the function $h : \mathbb{Z}^2 \to I$ is defined by

$$h * f(x,y) := \sum_{p,q\in\mathbb{Z}} h(p,q)f(x-p, y-q). \tag{2.1}$$

A change of variable in the sum shows that $h * f = f * h$. Thus, the action of a linear space-invariant operator L is the same as a *convolution with an impulse response function*, h, $Lf = h * f$.

Filter Implementation

It can be seen from the definition (2.1) that a filter is defined by a function h and that its effect on a pixel involves a weighted sum of the intensities over the whole image. For a real image, we shall assume that the support is an $M \times N$ rectangle, $f : B \to I$, where $B = [0, M-1] \times [0, N-1] \subset \mathbb{Z}^2$. For speed, and in order to minimize boundary effects the function h is often implemented through an associated function called the *kernel* or *mask* of the filter which has small support (3×3 matrices are extremely common).

Filtering can be performed in both spatial and the frequency domain. The pixel gray values at each location belong to the spatial domain. The range of frequencies obtained via the Fourier Transform of the image comprise the frequency domain of the image. A discussion on these two methods of filtering is given in the next section.

2.2 Spatial Domain Analysis of an Image

For $M \times N$ images a filter defined by a function $h(x, y)$ is given by the convolution

$$h * f(x, y) = \sum_{m,n \in B} h(m, n) f(x - m, y - n)$$

Kernels with support of small size are often defined in matrix form, and the action expressed as the scalar product with the same size matrix of pixels centered on a particular pixel.

In general, filter kernels of this type are defined by $(2J+1) \times (2J+1)$ matrices $w(p,q)$, $-J \leq p,q \leq J \subset \mathbb{Z}$. The corresponding filter function $h(x,y)$ is defined through a centralizing shift operator S, $(x,y) \rightarrow (x+J, y+J)$.

$$h(m,n) = S^{-1} w(J-m, J-n) S$$

for $0 \leq m,n \leq 2J$, and 0 otherwise.

Examples of such filters which are usually implemented with small support are the Sobel, Prewitt, and Roberts filters. Because of their compact support these filters achieve a considerable improvement in the implementation time compared with a filter which extends over the whole image ($O(mn(2J+1)^2)$ as opposed to $O(m^2n^2)$).

2.3 Frequency Domain Analysis of an Image

The gray levels of an image can also be manipulated by using filters that are defined in the frequency domain. The image has to be first Fourier transformed before a frequency domain filter can be applied, and then the transformed image is recovered by applying the inverse Fourier transform. The Fast Fourier Transform(FFT) can make the application of such a filter extremely rapid, particularly if the image is padded out with zeros so that its dimensions are powers of 2.

2.3.1 Image Representation via the Fourier Transform

The Discrete Fourier Transform(DFT) $F := \mathfrak{F}[f]$ of a function $f(x,y)$ representing an image of size $M \times N$ is given by the following expression:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2i\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

for $u = 0,\ldots,M-1$ and $v = 0,\ldots,N-1$.

The function $f(x,y)$ can be obtained via *the inverse Fourier Transform* of $F(u,v)$, $f = \mathfrak{F}^{-1}[F]$, as given by the following expression:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{2i\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

for $x = 0,\ldots,M-1$ and $y = 0,\ldots,N-1$.

The above two equations are together called the *two-dimensional Discrete Fourier Transform Pair.* The variables u and v are the transform or frequency variables, and x and y are the spatial or image variables.

2.3.2 Features of the Fourier Transform

1. The Fourier spectrum, phase angle, and power spectrum are

$$|F(u,v)| = \sqrt{\Re^2(u,v) + \Im^2(u,v)} \tag{2.2}$$

$$\phi(u,v) = \tan^{-1} \frac{\Im(u,v)}{\Re(u,v)} \tag{2.3}$$

$$P(u,v) = |F(u,v)|^2 = \Re^2(u,v) + \Im^2(u,v) \tag{2.4}$$

where $\Re(u,v)$ and $\Im(u,v)$ are the real and imaginary parts of $F(u,v)$, respectively.

2. One expression which is of interest in image analysis is the Fourier Transform of $(-1)^{x+y}f(x,y)$ shown below:

$$\mathfrak{F}[f(x,y)(-1)^{x+y}] = F\left(u - \frac{M}{2}, v - \frac{N}{2}\right).$$

Thus, by simply multiplying $f(x,y)$ by $(-1)^{x+y}$, the origin of $F(u,v)$ shifts to the frequency coordinates $\left(\frac{M}{2}, \frac{N}{2}\right)$. This is interpreted as $\left(\lfloor\frac{M}{2}\rfloor, \lfloor\frac{N}{2}\rfloor\right)$. This point is the center of the $M \times N$ area of the $2D$ DFT. The area of this frequency domain is called the frequency rectangle.

The value of the transform at $(u,v) = (0,0)$ is

$$F(0,0) = \frac{1}{MN}\sum_{x}^{M-1}\sum_{y}^{N-1} f(x,y)$$

which is the *average of* $f(x,y)$.

3. The Fourier Transform is conjugate symmetric for a real $f(x,y)$.

$$F(u,v) = F^*(-u,-v)$$

This implies that its spectrum is symmetric.

$$|F(u,v)| = |F(-u,-v)|$$

These properties of conjugate symmetry and centering find application in circularly symmetric filters in the frequency domain.

2.3.3 Relationship between Gray Level Values and the Frequency Domain

The above relations aid in dealing with measurements in the images during their processing. Regions of the image in which the intensity varies rapidly correspond to high frequencies in the Fourier transformed image, whereas the low frequencies are generated by regions in which the intensity varies slowly. It is, thus, the higher frequencies that indicate the presence of edges or noise in the image.

2.3.4 Filtering in the Frequency Domain

In 2.1 we saw that in the spatial domain a linear filter was equivalent to a convolution with an impulse function h, that is, $h * f$, where f is the image (2.1). In the frequency domain convolution becomes a multiplication if we first rescale the definition of convolution (otherwise a factor MN is introduced). Therefore, we henceforth use the following definition for convolution.

$$f * h(x,y) = \frac{1}{MN} \sum_{m,n} f(m,n) h(x-m, y-n)$$

$$\mathfrak{F}(f * h)(u,v) = \frac{1}{MN} \sum_{x,y} \left(\frac{1}{MN} \sum_{m,n} f(m,n) h(x-m, y-n) \right) e^{-2i\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}.$$

By applying a change of variable, i.e., $p{=}x-m$ and $q{=}y-n$, the above expression can be written as

$$\mathfrak{F}(f * h)(u,v) = \frac{1}{(MN)^2} \sum_{m,n} f(m,n) \sum_{p,q} h(p,q) e^{-2\pi i\left(\frac{u(p+m)}{M}+\frac{v(q+n)}{N}\right)}.$$

Rearranging and simplifying further gives the following result:

$$\mathfrak{F}(f * h)(u,v) = \frac{1}{(MN)^2} \sum_{m,n} f(m,n) e^{-2\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)} \sum_{p,q} h(p,q) e^{-2\pi i\left(\frac{up}{M}+\frac{vq}{N}\right)}.$$

The convolution in the frequency domain is then given by

$$\mathfrak{F}(f * h)(u,v) = [\mathfrak{F}(f)(u,v)][\mathfrak{F}(h)(u,v)].$$

Thus the Fourier transformed image is pointwise multiplied with an appropriate filter defined in the frequency domain or the Fourier transform of a spatial filter. The transformed image is then recovered by applying the inverse transformation.

To summarize, if $F(u,v)$ is the Fourier Transform of the image $f(x,y)$, and $H(u,v)$ a filter in the frequency domain, then the filtered transformed image, $G(u,v)$, is given by

$$G(u,v) = H(u,v)F(u,v).$$

The final filtered image, $g(x,y)$ is then obtained by taking the inverse Fourier Transform of $G(u,v)$,

$$g(x,y) = \mathfrak{F}^{-1}[G(u,v)].$$

2.3.5 Frequency Filters

A *low-pass* filter preserves low frequencies while weakening the higher ones by mapping them to zero. Low frequencies in the Fourier Transform capture the areas with constant gray levels with a lesser emphasis on the sharp details. Low pass filters in common use include the Gaussian lowpass, and Butterworth lowpass filters. For high values of a parameter called the filter order, the Butterworth filter behaves as an ideal filter while for corresponding lower values it behaves as the Gaussian filter. An *ideal low-pass* filter is one that cuts off all high frequency components of the Fourier Transform that are a distance greater than a specified cut-off distance from the origin while preserving those inside it. A two-dimensional ideal low-pass filter $H(x, y)$ in the $(u, v)-$plane is one which has compact support,

$$H(u,v) = \begin{cases} 1 \; if \; D(u,v) \le D_0 \\ 0 \; if \; D(u,v) > D_0 \end{cases}$$

where D_0 is the cutoff frequency, and $D(u, v)$=$\sqrt{u^2 + v^2}$.

A *high-pass* filter preserves high frequencies while removing the lower ones. High frequencies in the Fourier Transform capture the areas with a sharp transition in the gray levels like the edges and those due to noise while de-emphasizing the slowly varying gray level regions. Examples of a highpass filter are the Butterworth highpass, Gaussian highpass, and the Laplacian filter. A two-dimensional ideal high-pass filter is defined by

$$H(u,v) = \begin{cases} 0 \; if \; D(u,v) \le D_0 \\ 1 \; if \; D(u,v) > D_0 \end{cases}$$

where D_0, and $D(u, v)$ have the same description as given above. The filter has been arbitrarily normalized to 1.

Filters defined in the frequency domain can also be used to define kernels for spatial filters.

2.4 Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) provides a very efficient way of filtering images and is one reason why images are often filtered in the frequency domain. A one-dimensional

Fourier Transform of N points involves N multiplications N times. Thus, the total time is proportional to N^2. But a careful and a clever arrangement of these operations can optimize the algorithm to accomplish the same task in the order of $N \log_2 N$ operations. This optimized version is called the Fast Fourier Transform, FFT, or for the case of an image the Discrete Fast Fourier Transform, DFFT. The algorithm achieves this performance by a series of steps which decrease by powers of 2. The DFFT can be used in one of two ways. First the image is transformed to the frequency domain using the DFFT. For maximal efficiency it should first be padded out so that its dimensions are powers of 2. Then either the transformed filter is convolved with the transformed image or a filter defined in the frequency domain is convolved with it. The filtered image is then recovered by the inverse DFFT.

2.5 Image Processing Operations

In this section we shall primarily concentrate on discussing some well known filters for processing images.

2.5.1 The Gaussian Filter

Gaussian Functions have a bell-shape that is controlled by a parameter, σ, and have the property that their forward and inverse Fourier Transforms are both real Gaussian Functions. In the frequency domain, the *Gaussian filter function* of two variables is given by:

$$G(u,v) = e^{-2\pi^2\sigma^2(u^2+v^2)}$$

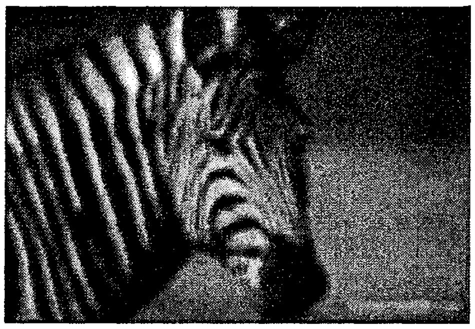
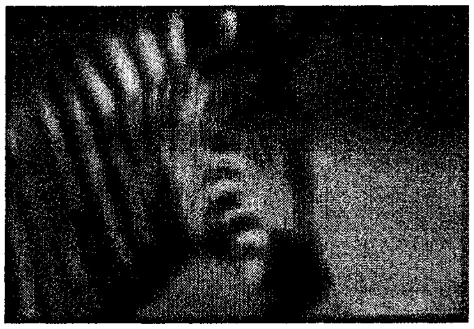
(a)  (b) 

Figure 2.1: The Zebra Smoothed Using a Gaussian Mask of (a)size 10×10 and σ=50 and (b)size 20×20 and σ=100

where σ is the standard deviation of the Gaussian Curve which measures its spread. The corresponding filter in the spatial domain is

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.5}$$

The Gaussian function is a very useful low pass filter in signal and image analysis. It plays a major role in smoothing an image and the subsequent noise reduction in it.

2.5.2 Normed Gradient Operation

Normed gradient filters enhance certain image features such as edges. They are high-pass filters which are sensitive to changes in the magnitude and the direction of the intensity in an image. For a function, $f : R^2 \to R^2$, the gradient of $f(x, y)$ at (x, y) is given by the two-dimensional vector

$$\nabla f(x, y) = (f_x, f_y)$$

where $f_x = \frac{\partial f}{\partial x}$, etc. The directional derivative of f along a curve

$$\mathbf{x} = (x = x(s), y = y(s))$$

if the curve is parameterized by its arc-length is

$$\frac{df}{ds} = \nabla f \cdot \frac{d\mathrm{x}}{ds} = \frac{\partial f}{\partial x}\frac{dx}{ds} + \frac{\partial f}{\partial y}\frac{dy}{ds} = f_x \cos\theta + f_y \sin\theta.$$

where θ=$\theta(x,y)$ is the angle between the tangent and the x-axis. The maximum value of $\frac{df}{ds}$ is obtained when $\frac{d}{d\theta}\left(\frac{df}{ds}\right)$=0. This gives

$$-f_x \sin\theta + f_y \cos\theta = 0 \Rightarrow \theta = \tan^{-1}\frac{f_y}{f_x} \tag{2.6}$$

$$\left(\frac{df}{ds}\right)_{max} = \sqrt{f_x^2 + f_y^2} \tag{2.7}$$

where θ is the angle for the maximum rate of change in f.

Therefore, ∇f points in the direction of maximum rate of change. The magnitude of this vector *which in the literature is often called the gradient*, $|\nabla f(x,y)|$ is used for edge detection,

$$|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2} \quad .$$

Gradient Direction

This term, which we used later in the thesis refers to the angle $\theta(x,y)$ which has just been introduced,

$$\theta(x,y) = \tan^{-1}\left(\frac{f_y}{f_x}\right).$$

Normed Gradient Approximation by Differencing

There are many different discrete versions of the normed gradient based upon difference schemes. For example, at the discrete site (x,y) we can represent $|\nabla f(x,y)|$ by using

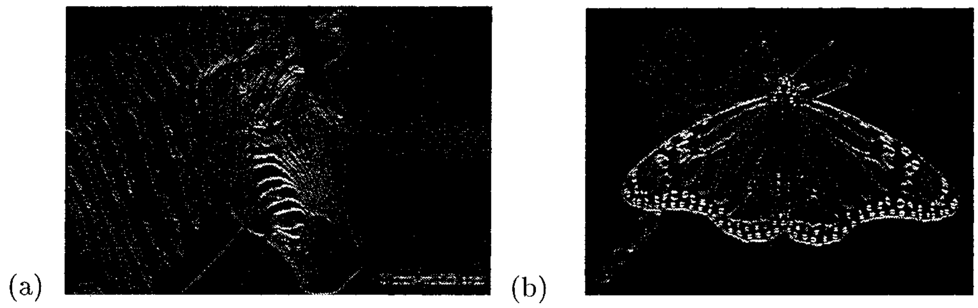


Figure 2.2: Normed Gradient of the (a)Zebra and the (b)Butterfly Images

first-order forward differencing in a 2×2 region. In the Euclidean norm it is given by

$$|\nabla f(x,y)| \cong \sqrt{[f(x+1,y)-f(x,y)]^2+[f(x,y+1)-f(x,y)]^2}.$$

Various approximations to the square root are also used. For example, the diamond norm uses only the absolute values

$$|\nabla f(x,y)| \cong |f(x+1,y)-f(x,y)|+|f(x,y+1)-f(x,y)|.$$

An approximation based on the diagonal pixels is also used

$$2|\nabla f(x,y)| \cong |f(x,y)-f(x+1,y+1)|+|f(x+1,y)-f(x,y+1)|.$$

Thus, in all approximations the value of the normed gradient depends on the difference in the gray level between the adjacent pixels. The normed gradient is then expected to assume large values for prominent edges in an image, small values in regions that are fairly smooth, and zero in regions with constant gray level.

2.5.3 Laplacian Filter

This is another high-pass filter used in edge detection. A Laplacian is zero at points where the normed gradient is a maximum or a minimum, and so gradient edges will go undetected by it. The Laplacian is

$$\nabla^2 f = f_{xx} + f_{yy}.$$

The discrete version of the Laplacian obtained by central differencing is

$$\begin{aligned}
f_{xx} &\cong f(x+1,y) + f(x-1,y) - 2f(x,y) \\
f_{yy} &\cong f(x,y+1) + f(x,y-1) - 2f(x,y) \\
\nabla^2 f &\cong f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)
\end{aligned}$$

The coefficient matrix for $\nabla^2 f$ defines the following mask

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Edge detection using the Laplace operator

The normed gradient operation works best when the gray-level transition is quite abrupt, as is in the case of a step function. But as the transition region gets wider, it is more advantageous to apply the Laplacian. The problem with the Laplacian is that it produces double edges; it is also sensitive to noise. For these reasons together with its inability to detect the edge direction, the Laplacian if used alone is not a good edge detector. Zero crossings provide a way around this.

The detection of the *zero-crossings* of the Laplacian provides another way of locating edges. Because of the sensitivity of the Laplacian to noise the image is first smoothed before

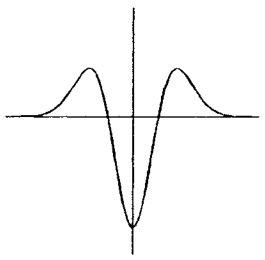


Figure 2.3: The Mexican Hat Wavelet or the LOG Filter

the Laplacian is applied. As discussed in the original paper on zero-crossings [25], the low-pass filter should be local in the spatial domain. The Gaussian filter $g(x, y)$ introduced earlier (2.5) has exactly the required properties. The combined action of the Laplacian and the Gaussian filter leads to the *Laplacian of the Gaussian*, (LOG) filter. Its derivation is discussed in the next subsection. The discrete version of this filter is given by

$$h(m,n) = \frac{1}{2\pi\sigma^6}(m^2 + n^2 - 2\sigma^2)\exp\left(-\frac{m^2+n^2}{2\sigma^2}\right)$$

The action of the LOG filter depends strongly on the variance σ. Larger values of σ increase the smoothing and reduce the number of zero-crossings that can be located. Pixels which effectively contribute to the smoothing are located within a distance of 3σ from the current pixel.

The LOG filter is also referred to as the *Mexican hat wavelet*. It is a well known example of a continuous wavelet, see chapter 4. The LOG filter thus provides a natural way of analyzing an image at different scales controlled by the σ parameter.

There are several ways in which the zero-crossings can be detected. A simple way is to use a threshold of zero so that a binary image is obtained. The results of using the LOG

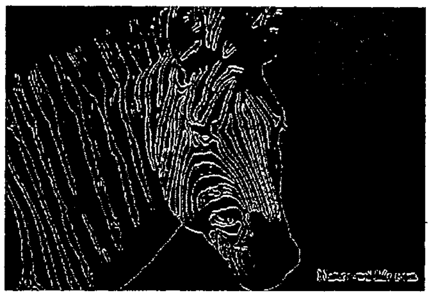
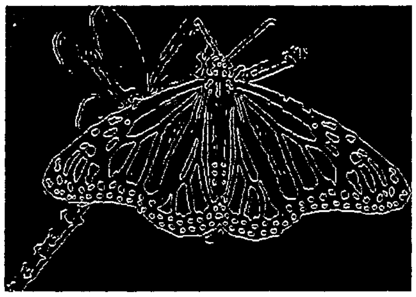
(a)  (b) 

Figure 2.4: LOG Edges of the (a)Zebra and the (b)Butterfly Images

as the filter for edge detection can be seen on the zebra and the butterfly, 2.5.3.

2.5.4 Differentiable Filters

In the case of a differentiable filter, the action of the filter and a differential operator on an image can be combined into a new filter. Suppose that $s=s(x,y)$ is a differentiable filter. The action of the filter on the image f is given by $f * s$. Let ∂_x also represent the action of the derivative operator ∂_x on f. For the case of a differentiable function f we have

$$\begin{aligned}(\partial_x f * s)(x,y) &= \int\int_{supp(f)} \partial_x f(x-p,y-q)s(p,q)dpdq \\ &= -\int\int_{supp(f)} \partial_p f(x-p,y-q)s(p,q)dpdq \\ &= -\int f(x-p,y-q)s(p,q)dq|_{p\in\partial supp(f)} \\ &\quad + \int\int_{supp(f)} f(x-p,y-q)\partial_p s(p,q)dpdq\end{aligned}$$

Thus if we make the assumption that s vanishes on the boundary of the image $supp(f)$, then

$$\partial_x f * s = f * \partial_x s.$$

We take this to define the action of ∂_x on an image convolved with a differentiable filter s. In particular this result gives

$$\begin{aligned} \nabla f * s &= f * \nabla s \\ \nabla^2 f * s &= f * \nabla^2 s. \end{aligned} \tag{2.8}$$

For the Gaussian filter the operation of Gaussian smoothing followed by the Laplacian is equivalent to the action of the LOG filter

$$\nabla^2 g(x,y) = \frac{1}{2\pi\sigma^6}(x^2+y^2-2\sigma^2)\exp^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{2.9}$$

2.6 Some Special Filters

Implementations of standard filters are often named after their inventors. Any implementation has its strong and weak points. Here we discuss the implementations of the normed gradient filter which are common in use.

2.6.1 Sobel Filter

The Sobel filter uses first and second order differencing to approximate the first order derivatives,

$$\begin{aligned} 8f_x &\cong f(x+1,y+1)+2f(x+1,y)+f(x+1,y-1) && (2.10)\\ &\quad -(f(x-1,y+1)+2f(x-1,y)+f(x-1,y-1)) \\ 8f_x &\cong f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1) && (2.11)\\ &\quad -(f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)). \end{aligned}$$

The normed gradient is determined using the diamond norm, but in the implementation it is $8|\nabla f|$ which is calculated. This saves a division.

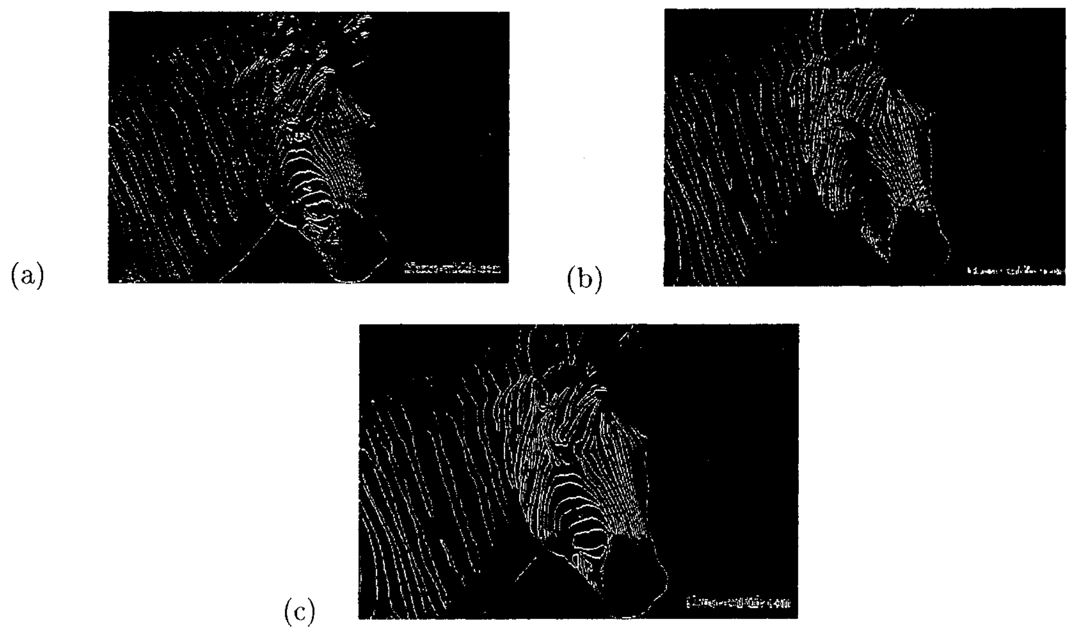


Figure 2.5: Sobel Operations on the Zebra Image: (a) f_x, (b) f_y, and (c) Sobel Edges

The masks for the derivative operators are

$$h_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad h_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}.$$

Notice that h_y is the transpose of h_x, but defines $-8f_y$, though this doesn't matter if the normed gradient is being calculated. One disadvantage of using the Sobel filter is that it can exaggerate and thicken edges associated with noise.

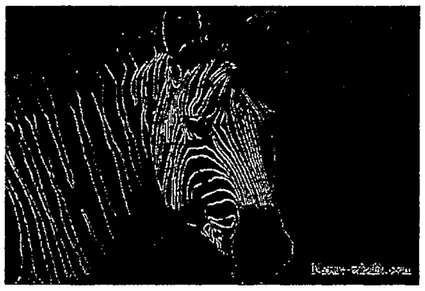
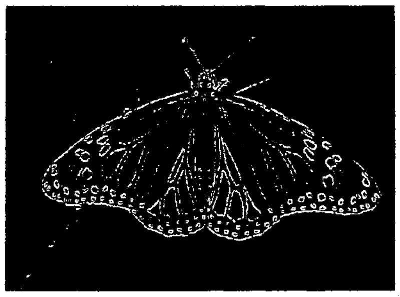
(a)  (b) 

Figure 2.6: Roberts Edges of the (a)Zebra and (b)Butterfly Images

2.6.2 Roberts Filter

This filter for the normed gradient was mentioned in section 2.5.2,

$$2|\nabla f| \cong |f(x,y) - f(x+1,y+1)| + |f(x+1,y) - f(x,y+1)|.$$

This expression follows from the relation

$$2(f_x^2 + f_y^2) = (f_x - f_y)^2 + (f_x + f_y)^2$$

and then using the diamond norm for the vector $(-(f_x + f_y), (f_x - f_y))$. The masks h_1, h_2 for the differences $f(x,y) - f(x+1,y+1)$ and $f(x+1,y) - f(x,y+1)$ are

$$h_1 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad h_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

where in both cases the lower left corner corresponds to the current pixel. It can be seen from the masks or equivalently the Robert's filter that edges at $\pm\frac{\pi}{4}$ to the x-axis are most clearly detected. The accompanying figures involving the use of Robert's filter support this assertion.

2.7 Thresholding

Thresholding is a method of dividing pixel values into different classes, and then assigning white say to pixels in certain selected classes and black to the remaining pixels. The output is therefore a binary image.

Thresholding is used to distinguish an object in the image from the background. It will clearly be most effective when the gray scale of the object is very different from the background. For example, black writing on white paper. However, sophisticated methods have been developed to handle situations when there is variation in the gray scale of the object as well as its background. A recent survey, [26], gives a detailed comparison of the effectiveness of the different methods.

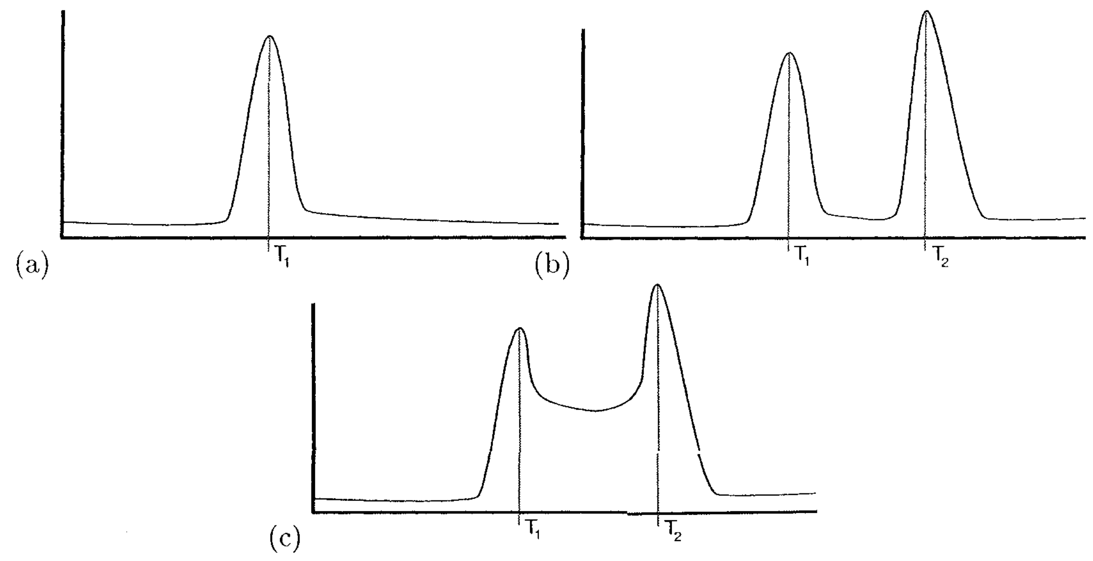


Figure 2.7: Thresholding Operations on the Zebra Image: (a) Single, (b) Two, and (c) Multiple Thresholds

In the simplest implementation a single threshold T_1 is chosen and pixels with intensities higher than T_1 are set to white in the output, whereas pixels of lower intensity are set to black. The applicability of the method to a given image can be deduced from the histogram of intensity values. Figure (2.7) shows some of the possible cases. For case (a) the object, assumed to have low gray scale values, is clearly delineated from the background and can be selected by thresholding with the value T_1. In figure (b) the object is represented by the central maximum and again can be isolated by setting pixels with intensities less than T_1, or greater than T_2 to white. The third diagram indicates a case where simple thresholding will not work, as there is considerable overlap between the gray scales of the object and the background. In this case an iterative cluster-based threshold method might be effective. At each stage pixels are divided into two clusters which correspond to the two maxima of the intensity histogram. The threshold is defined to be the mean T_m of the two maxima and the pixels are thresholded so that pixels with intensities greater than T_m are set to white, thus reducing the number of pixels to be thresholded at the next level. The process is terminated when the difference between the locations of the two maxima $|T_1 - T_2|$ is decided to be sufficiently small.

Thresholding can also be used to modify the output from other filters. For example, it can be used with an edge detector so that only the finest edges are retained.

2.8 The Canny Edge Detector

The edge detectors discussed so far suffer from several problems such as not locating all edges or conversely detecting edges where none exist. More sophisticated edge detectors are based on first establishing criteria that the detector is to meet and then mathematically

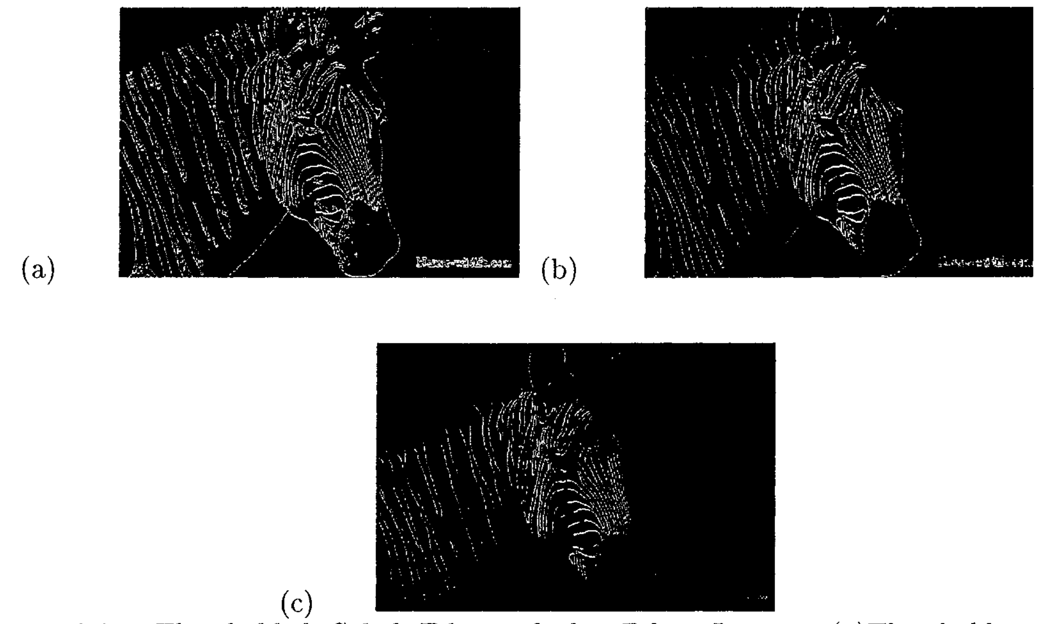


Figure 2.8: Thresholded Sobel Edges of the Zebra Image: (a)Threshold = 0.1, (b)Threshold = 0.2, and (c)Threshold = 0.3, where edges stronger than the threshold value are not retained.

analyzing them to derive the detector.

There are several detectors of this type. The first is due to Canny [21]. Others that we should mention are due to Shen and Casten [60], and Smith and Brady [28]. In this thesis we only discuss the Canny detector. One of the reasons for this is because it can be used to detect edges at different scales and therefore fits in neatly with the approach that we are using for the Watershed Transformation.

The Canny edge detector consists of several algorithms which locate edge pixels and then attempt to link them together into edges. First though we discuss Canny's approach to detecting the direction of an edge. At an edge pixel ∇f is normal to the edge, as seen in §2.6. Therefore an edge can be detected by considering the line $\mathbf{x}=\mathbf{x}_0 + \lambda\nabla f|_{\mathbf{x}_0}$, where $\mathbf{x}_0$ is a given point in the image. The point $\mathbf{x}_0$ is an edge point if $|\nabla f|_{\mathbf{x}}$ is at a maximum at $x=x_0$. Note that the edge points are inflection points of f. The angle of the edge direction is given by $tan^{-1}(-\frac{f_y}{f_x})$. The normed gradient $|\nabla f|$ is sometimes calculated using the diamond norm $|\nabla f|_\diamond = |f_x| + |f_y|$ instead of the Euclidean norm.

2.8.1 Cannys Criteria

These are:

- Optimum edge pixel detection so that all edges are found and the number of false edges is minimized
- Obtain thin edges by forming them as close as possible to the center of the true edge
- Suppress multiple edge generation when only a single edge is present

Canny converts the criteria into a total error cost function. Variational calculus is then used to minimize the cost function to define an optimal linear operator which best meets the criteria. It turns out that the first derivative of the Gaussian is very close to the optimal operator and this what is used in the detector.

A novel feature of the algorithm is the use of *hysteresis thresholding* to reduce streaking in the output which occurs if only a single threshold is used; for an edge can only vary in intensity and parts of the edge below the threshold will be lost. Hysteresis is a phenomenon in which the response of a physical system to an external influence depends not only on the present magnitude of that influence, but also on the previous history of the system.

2.8.2 Cannys Algorithm

As usually implemented this involves a multi-stage process which is divided into the following routines.

1. *Smooth the Image*

   The image is smoothed with a Gaussian filter. This reduces the number of false edges, but thickens the edges and so increases the error in the localization of the true edge.

2. *Compute the Gradient of the Smoothed Image*

   The derivatives are calculated using forward differences.

3. *Form the Normed Gradient Image*

   This is computed from the gradient using either the Euclidean or the diamond norm. The edge directions are also calculated.

4. *Non-Maximal Suppression*

   This step reduces wide edges to thin edges in the gradient image. This is achieved by the folllowing action; for each edge pixel trace along the edge in the edge direction and zero any pixel whose normed gradient value is not a maximum.

5. *Hysteresis Thresholding*

   Two thresholds T_1, and T_2 are selected, $T_1 < T_2$. All magnitudes below T_1 are set to zero and made non-edges, while those above T_2 are made edges. For those pixels whose magnitudes lie between these two thresholds the possibility of a path from them to a pixel having a value above T_2 is explored. If such a path is not found then it is made a non-edge.

2.8.3 Implementation Details

The signal to noise ratio is affected by the value of σ. Basically, the larger the width of the Gaussian the lower the detector's sensitivity to noise. For sharp edges though σ needs to be fairly small, so there is a trade-off between these two situations. In many implementations σ is not a parameter of the algorithm and the Gaussian is implemented as a convolution mask with a fixed value of σ.

The gradient of the image is often determined using the h_x, h_y masks of the Sobel operator, §2.6.1. The edge direction is then given by $-\tan^{-1}(\frac{h_x}{h_y})$. Since the possible edge directions on an image are multiples of $\frac{\pi}{4}$, the algorithm selects the value nearest to the real value as the edge direction.

2.8.4 Edge Detection at Multiple Scales

In Canny's paper [21] the edge detector he describes is considerably more sophisticated than the one we have presented. If $g_\sigma(x, y)$ denotes the Gaussian with standard deviation σ, then the gradient of the Gaussian should be implemented as in §2.5.4. Thus for the image f it is given by

$$\nabla f_\sigma(x, y) = (f * \nabla g_\sigma)(x, y).$$

Then the gradient magnitude in the Euclidean norm is

$$|\nabla f_\sigma| = \sqrt{(\partial_x f_\sigma)^2 + (\partial_y f_\sigma)^2}$$

and the direction of the local maxima is given by $\tan^{-1}\frac{\partial_y f_\sigma(x,y)}{\partial_x f_\sigma(x,y)}$.

The detector is therefore naturally set up to detect edges at different scales. Canny suggests that the scale should be automatically adjusted by an estimate of the signal-to-noise ratio as the detector processes the image. this thesis we prefer to treat the scale as a parameter of the detector. In this way we can compare the output from the edge detector at different scales with the output from the Watershed Transformation at the same scale.

A further problem that can be investigated for both the scaled outputs of the edge detector and the Watershed Transformation is how best to combine the outputs to give the most information about the edges or contours. For example the outputs can be added to combine the edges at different scales or intersected to give the edges that are common across a set of scales. Canny introduces a third possibility, that of *feature synthesis*. In this approach all the edges are first determined for an image at a fine scale. These edges are then used to synthesize the output at a coarser scale by convolution with a Gaussian of larger σ. A comparison is then made with the edges which appear in the output of the detector at this coarser scale. Edges are only added into the original set if they have a considerably greater response than that which occurs in the synthesized output.

2.8.5 Some Examples

Canny edges can be seen in the Zebra, and the Butterfly. The edges identified by the canny operator are extremely good in comparison to those identified by the Watershed Transform, which is discussed in the next chapter.

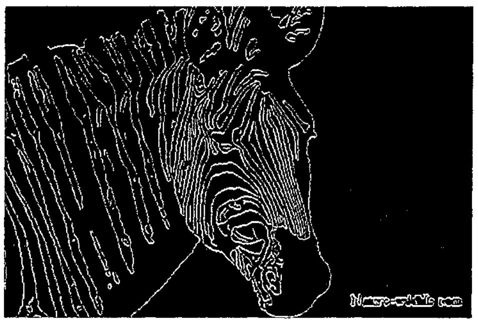
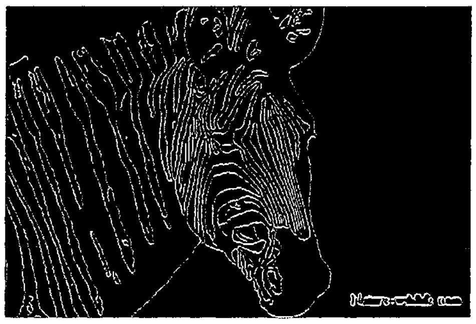
(a)  (b) 

Figure 2.9: Canny Edges of the Zebra Image at (a) $\sigma = 0.5$, and (b) $\sigma = 50.5$.

Chapter 3

The Watershed Transformation

3.1 Introduction

The Watershed Transform is a segmentation procedure to obtain the contours in an image. It is one of the few methods which extracts closed boundaries by operating directly on the pixels in a grayscale image. There are basically two ways of extracting watersheds, one which uses the idea of rain falling on a landscape and the other which works on the idea of immersing a landscape in a lake. Since the two methods work on different sets of neighborhood pixels it is not surprising that the watersheds obtained from them are not the same. This chapter discusses these two ideas and their mathematical basis, and also their implementation.

One major disadvantage of this algorithm is that oversegmentation occurs. This can be seen in figure 3.1 where the generation of contours due to noise obscures the contours related to the ferns.

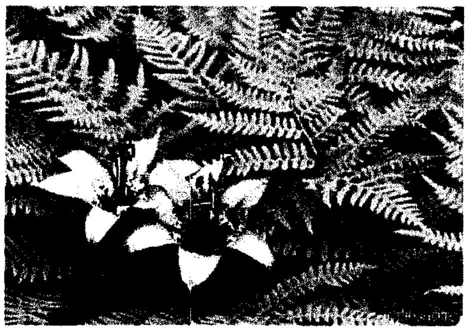
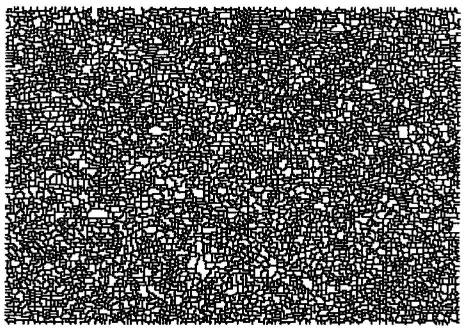
(a)  (b) 

Figure 3.1: Immersion Watersheds (b) for the Ferns Image (a)

3.1.1 The Topographical Analogy

Every image can be translated into a topography or a landscape where the height of the hills corresponds to the higher, and the depth of the valleys to the lower pixel intensities. Thus, if a grayscale image is viewed as if high intensity pixels were high ground and low intensity pixels were low ground then the image becomes a 3D landscape. A minimum is then a dip in the ground surrounded by higher land. Under the rainfalling analogy, the catchment basin of such a minimum is the area where water falling on the landscape would flow down to this minimum. Under the immersion analogy, catchment basins around each minima are formed when water coming from the corresponding minimum progressively fills it up. The watershed of the image is then the set of curves that separate the catchment basins of the image. The watershed contours can be quite thick and extreme cases are called plateaus. Special techniques have been developed to thin them.

If a Watershed Transform is applied directly to the image it creates watersheds at all the pixels with the largest gray value. But for better segmentation results, watersheds should be created at pixels where the gray value changes the most. Methods like taking the normed gradient of an image aid in identifying such pixels. Some other methods like smoothing the image can help reduce the appearance of false edges due to noise present in an image. Thus, such preprocessing of an image can reduce the number of spurious watersheds.

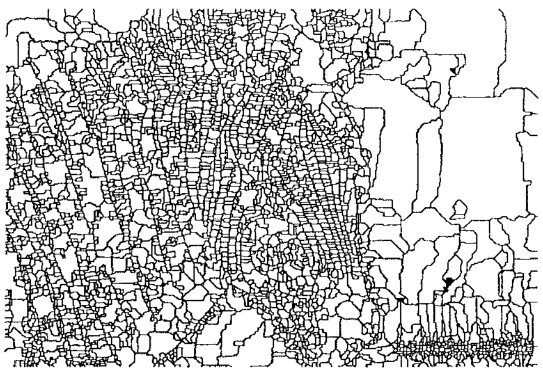


Figure 3.2: 4-connectivity Immersion-based Watershed Transform of the Zebra

3.2 Definitions

It is convenient to group together all the definitions for both methods before we discuss them in detail. An excellent reference for this method is the review article by [1].

General Graph Properties:

- A graph $G=(V, E)$ consists of a set V of *vertices* and a set $E \subseteq V \times V$ of *edges*. If the graph is undirected then if $(v_1, v_2) \in E$ then (v_2, v_1) also represents the same edge. In a *directed graph* the edge $(v_1, v_2) \in E$ is distinct from the edge (v_2, v_1) (if it exists).

- A path p of length l consists of a sequence of vertices $\{v_1, v_2, \cdots, v_l\} \in V$ such that

$(v_i, v_{i+1}) \in E$, $i = 1, \cdots, l-1$. We denote this path by p=$(v_1, \cdots, v_l)$ and write length(p)=l.

- If $v, \tilde{v} \in V$ then $\tilde{v}$ is *reachable* from v, written $v \leadsto \tilde{v}$, if there exists a path from v to $\tilde{v}$. A graph G is *connected* if every pair of vertices in G are path connected or alternatively every vertex is reachable from any other vertex. A *connected component* of G is a maximal connected sub-graph of G.

- A path p=$(v_1, \cdots, v_l)$ is a *cycle* if $v_1 = v_l$ and the $v_j, 1 \leq j \leq l-1$ are distinct. A graph without any cycles is *acyclic* or a *forest.*

- A *tree* is a connected acyclic graph.

Properties for the Digital Gray Scale Image:

- A graph G=(V, E) where $V \subseteq \mathbb{Z}^2$ constitutes the support for an image. The vertices in this case are called pixels. The definition of an edge depends upon the *connectivity.* In 4-*connectivity* the edges associated with a pixel are the edges to the horizontal and the vertical neighbors. In 8-*connectivity* the edges to its diagonal neighbors are also included.

- A digital image is a triple G=(V, E, f), where $V \subseteq \mathbb{Z}^2$ is the support of the image with edges defined by $4-$ or $8-$ connectivity. The function $f : V \to \mathbb{Z}_+$ assigns a gray value to each pixel in the image. Often the values for f are scaled into the standard 8 bit gray scale values $0 - 255$. An image which has the two values 0 or 1 is called a binary image.

- A *level component at level h* or *plateau* of an image is a connected component of the image for which the pixels v all have the same gray scale value, $f(v){=}h$. The *boundary* of a level h component, C, of an image consists of the pixels $v \in C$ that have one or more neighbors $\tilde{v}$ for which $f(v) \neq h$. A *local minimum* at level h of an image is a level h component C of the image such that if $\tilde{v}$ is any neighboring pixel of a boundary pixel, $f(\tilde{v}) \geq h$.

- A *descending path* $p{=}(v_1, \cdots, v_l)$ is a path such that $f(v_1) \geq f(v_2) \geq \cdots \geq f(v_l)$ and $f(v_l) < f(v_1)$. Let $\pi_f^{\downarrow}(v)$ denote the set of all descending paths starting at v. If C itself is a local minimum then for $v \in C$, $\pi_f^{\downarrow}(v){=}\emptyset$. An image is *lower complete* if every vertex v which is not a minimum has a neighbor $\tilde{v}$ for which $f(\tilde{v}) < f(v)$.

Topological Definitions Specific to Watershed Transform:

Let $G{=}(V, E, f)$ be a gray scale image.

- The *threshold set*, of f at level h is

$$T_h = \{v \in V : f(v) \leqslant h\}.$$

- For a sub-image $A \subseteq G$ the *geodesic distance* between pixels $a, b \in A$ is the minimal path length of all paths which connect a and b, and lie entirely in A.

- If $B \subseteq A$, define

$$d_A(a, B) = \min_{b \in B}(d_A(a, b)).$$

Let $B \subseteq A$ be partitioned into k connected components B_i, $i{=}1, \ldots, k$.

- The *geodesic influence zone*, of B_i within A is defined as

$$iz_A(B_i) = \{v \in A : \text{for all } 1 \leq j \leq k, j \neq i, d_A(v, B_i) < d_A(v, B_j)\}.$$

- The *union of the geodesic influence zones* $IZ_A(B)$ is the geodesic influence zones of the connected components of B, given by

$$IZ_A(B) = \bigcup_{i=1}^{k} iz_A(B_i).$$

- The *skeleton of influence zones* $SKIZ_A(B)$ is the complement of the set $IZ_A(B)$ within A.

$$SKIZ_A(B) = A \setminus IZ_A(B)$$

  The $SKIZ$ contains all points which are equidistant, w.r.t. the geodesic distance, to at least two nearest connected components (none for digital grids). For a binary image f with domain A, $SKIZ$ can be defined by identifying set B with the set of pixels having the highest gray value, i.e 1.

3.2.1 Definition of the Watershed Transform

The Watersheds are a skeleton by influence zones determined in relation to geodesic distances. As described in [5] these can be viewed as a generalization of the skeleton of influence zones ($SKIZ$) to gray value images.

Definition: Let $f : D_I \to \mathbb{Z}$ have minima m_k, $k \in I$ for some index set I. The catchment basin $B(m_i)$ of a minimum m_i is defined as the set of points $x \in D$ which are topographically closer to m_i than to any other regional minimum m_j.

$$B(m_i) = \{x \in D | \text{for all } j \in I \setminus i : f(m_i) + T_f(x, m_i) < f(m_j) + T_f(x, m_j)\}$$

The *Watershed* of f is the set of points which do not belong to any catchment basin.

$$Wshed(f) = D \cap \{\bigcup_{i=1} B(m_i)\}^c$$

Let W be some label, $W \notin I$. The watershed transform of f is a mapping $\lambda : D \to I \cup W$ such that if

$$p \in B(m_i) \Rightarrow \lambda(p) = i \tag{3.1}$$

$$p \in Wshed(f) \Rightarrow \lambda(p) = W \tag{3.2}$$

Thus, the watershed transform of f labels all points of D either uniquely for each catchment basin or as a special symbol W for all points of the watershed of f.

3.2.2 Occurence of Plateaus

A topographic relief can contain discontinuous structures representing thick watersheds, namely, plateaus. These are regions of constant gray values. Their existence in an image is a main obstacle when one tries to extend the Watershed definition from the continuous case to the discrete case. Thus, we need an algorithmic definition which computes the Watershed transform level by level, where each level constitutes a binary image for which a *SKIZ* is computed.

3.3 Watershed Transform

The Watershed Transform produces a complete division of an image into separate regions, and generates closed contours. For determining the watersheds one needs to categorize or label all the pixels as belonging to either the watershed lines, catchment basins and/or minima. The identification of the pixels can be done in two ways:

- *rainfalling/tobogganing method*
- *immersion/flooding method*

The details of these two are discussed in the following sections.

The various stages involved in isolating the watershed pixels can be summarized as:

- *Identifying* the watershed pixels depends on the underlying method used for isolating them from the background. For this one needs a quantity that takes a high value in the neighborhood of edges and low values for the interior pixels. Methods like the normed gradient, or teager energy detection [20], can achieve this.

- *Preserving* an optimum number of pixels which need to be retained in the image to get an acceptable segmentation.

- *Improving accuracy* of results by using methods like calculating the geodesic distances to determine more precisely the location of neighborhood pixels on a watershed.

3.4 The Rainfalling/Tobogganing Method

In this method, the path of a raindrop is traced using the idea of steepest descents, wherein one is interested in those pixels which are traversed by the raindrop in its descent. This raindrop will stop its descent if it reaches a minima or a plateau in the landscape. The descent continues under gravity and sometimes special points called saddle points are also encountered in this path. It is this sequence of points which defines the path of steepest descent that one is interested in extracting. The various steps involved in identifying such pixels can be enumerated as below:

1. *Removing the weakest edges*[10] which involves grouping all pixels having intensities below a certain threshold into one. This will lead to creation of lakes on the relief which would then represent the underground water level. This reduces oversegmentation mainly due to noise.

2. *Determining the steepest descent* for each pixel which involves determining the direction in which a raindrop would flow to if it were to fall on the topographic relief. The following are the possible cases:

    - If a pixel has only one steepest descent neighbor, merge the current pixel and this neighbor into the same linked list.
    - If a pixel has more than one steepest descent neighbor, then make this pixel belong to the watershed.
    - Catchment basins can be created by labelling all the lakes in step 2 and adding this label to pixel information of those pixels having only one steepest descent neighbor. This way the pixels can be related to the catchment basins.

Rainfalling/Tobogganing is based on the idea of the drainage analogy [34] wherein any two points are in the same region or catchment basin if they drain to the same minimum point. Since the Watershed Transform works best on the gradient image, we assume all pixels that need to be processed for the watersheds belong to the gradient image. To determine the drainage behavior of a pixel in gradient image one needs to find its downstream path along the steepest descent that ends in a local minimum of the topography. Thus, all pixels will either belong to a minima or the watershed of the gradient image.

Advantages of this Method

This method has the following advantages:

- It is non-iterative as every pixel is processed only once.
- Its execution time is linear.
- It keeps track of a single parameter that determines a neighboring pixel of strictly smallest gradient magnitude for every pixel. This process terminates if the path encounters an already processed pixel or reaches a local minimum.

3.4.1 Mathematical Representation of the Watershed Lines Using the Rainfalling Analogy

This section outlines the mathematical background for exploring all possible downhill paths from a pixel and determining the one that corresponds to the steepest descent. This path is followed until a regional minima is reached. The set of all pixels encountered on this path belongs to the catchment basin for this minimum. Repeating this process for all the pixels generates the set of all the catchment basins. The watershed lines are then the contours that separate the basins, so that a raindrop that falls on a watershed can flow to more than one minima. Terms like lower complete, lower slope, downstream, cost of walking, etc are defined below:

- The *lower slope* is the maximal slope linking a pixel p to any of its neighbors of lower altitude is defined as

$$LS(p) = MAX_{q \in p \cup N_G(p)} \frac{f(p) - f(q)}{d(p,q)}$$

- The *cost of walking* from one position p to a neighboring position q is defined as

$$cost(p,q) = \left\{ \begin{array}{ll} LS(p).d(p,q) & \text{if } f(p) > f(q) \\ LS(q).d(p,q) & \text{if } f(p) < f(q) \\ \frac{1}{2}(LS(p) + LS(q)).d(p,q) & \text{if } f(p) = f(q) \end{array} \right\}$$

- A pixel belongs to the *downstream* of a pixel p if there exists a path π of steepest decent between p and q.

Definition Using the Local Minima

Let M=$m_1, m_2, \cdots, m_k$ be the set of all minima of all local minima of f. The Watershed Transform is a labeling process that maps every point in D to a label domain W, where W=$1, 2, \cdots, k, u$, and u is a constant differing fom $1, 2, \cdots, k$. An algorithmic Watershed Transform can then be defined as

1. if p=m_i, $Wshed(p)$=i,
2. if $downstreampath(p)$ is undefined, $Wshed(p)$=u,

3. if $p \neq m_i$, $Wshed(p)$=$Wshed(downstreampath(p))$, for a selected downhill path $downstreampath(p)$.

Given above is a recursive definition of $Wshed(p)$. The labeled regions $L_1, L_2, \cdots, L_k, L_u$ are then generated, where L_i=$p \in D|Wshed(p)$=i, i=$1, 2, \cdots, k, u$. $Wshed(p)$ defined here is a labeling function rather than being one that finds the watershed lines which in turn are obtained by extracting the boundaries of all the labeled regions. The labeled regions L_i are connected for i=$1, 2, \cdots, k$; but the region L_u is not necessarily connected. It consists of the points that do not follow their downhill path to any local minima of f. This occurs because of the presence of plateaus in f. Thus, L_u consists of points that are in a plateau or whose downhill path ends up in a plateau.

3.4.2 The Idea of Steepest Descents and Watershed Lines

Every non-critical point on the watershed represents a saddle point. Using the rain falling analogy, the raindrops falling on it have equal probability of flowing to the valleys on either side.

3.4.3 A Saddle Point

A saddle point is a point at which a function of two variables has partial derivatives equal to zero but at which the function has neither a maximum nor a minimum value. Such a point is a stationary point but not an local extremum, and the surface resembles a saddle that curves up in one direction, and curves down in a different direction. In terms of contour lines, a saddle point can be recognized by a contour that appears to intersect itself. For example, two hills separated by a high pass will show up a saddle point, at the top of the pass, like a figure-eight contour line [18]. The Watershed pixels behave as saddle points. Thus, the saddle points are stable for catchment basins.

3.5 The Immersion Method

If we pierce holes at every regional minima and immerse the image surface into water, then water will start to flood areas adjacent to regional minima. A regional minimum is a connected plateau from which it is impossible to reach a point of lower gray value level by an always descending path. As the image surface is immersed, some of the flooded areas or the catchment basins will tend to merge. When two or more different flooded areas are touched, infinitely tall dams or watershed lines are constructed between them. When the water level reaches the highest peak in the landscape, the process is stopped. The resulting network of dams then defines the watershed of the image. In other words, watershed lines partition the image into nonintersecting patches, called the catchment basins. Since each patch contains only one regional minimum the number of patches is equal to the number of the regional minima in the image.

3.5.1 Mathematical Representation of the Watershed Lines Using Immersion Analogy

The general properties of the graph theory and the digital images are listed in section 3.2.2 lend meaning and support to the immersion-based Watershed Transform. The definitions pertaining to the geodesic distances are used to formally identify and extract the watersheds. The research for these ideas is mostly done from [1], [50], and [2].

The following ideas put together help define a recursive definition for the watersheds using the immersion method.

The *threshold set* for g at level ν is given by

$$S_v = \{p \in D | g(p) \leq \nu\}.$$

The *local intensity minima set* for f at level ν is given by

$$LIM_\nu = \{p \in D | g(p) = \nu, g(p) < g(q), \forall g \in N_n(p) \setminus \{p\}\}.$$

Let ν_{min} and ν_{max} be the minimum and maximum intensity value of gray scaled f. Then one can define a recursion with the value of g growing from ν_{min} to ν_{max}, as shown below.

$\mathrm{I}_{\nu_{min}} = \{p \in D | g(p) = \nu_{min}\}(3.3)$

$$I_\nu = LIM_\nu \cup IZ_{S\nu}(I_{\nu-1}), \nu = \nu_{min} + 1, \nu_{min} + 2, \cdots, \nu_{max} \tag{3.4}$$

The watersheds of g are then the complement of $I_\nu max$ as given by

$$Wshed(g) = D \setminus I_{\nu_{max}}.$$

The $Wshed$ defined here is a set of watershed points unlike the definitions in the previous section as it is a labeling process mapping all points in D into a set of labels.

3.6 Implementation Analysis for Extracting Watersheds

The following are true for the basins and watershed lines:

- All points of a given catchment basin have the same unique label
- A special label is given to all points of watersheds

3.6.1 The Rainfalling Case

Following are the steps for finding the watersheds using the *R*ainfalling analogy:

1. Compute the sets of neighborhood pixels for all pixels
2. Then make connected sets for each pixel depending upon the 4 or 8-connectivity used
3. Assign to the basin if a pixel lies along the path of steepest descent of the pixel
4. Assign to the watershed if the pixel has more than one steepest path of descent

3.6.2 Immersion/Flooding Case

Following are the steps for finding the watersheds using the *Immersion/Flooding* analogy:

1. Sort the pixels according to their gray values in an increasing order
2. Proceed with integer increments from the lowest gray value and do the following:
   (a) identify all minima at the current level

Table 3.1: Matlab Pseudo-Code for Sorting the Pixels for the Rainfalling Watersheds

```
I = imread('c:\zebraedges\zebra','jpg');
I = rgb2gray(I);
[m n]= size(I);

Sort the pixel positions int terms of their gray values in ascending order
pix_sort = [-1 -1 -1];
for i = 0:1:255
     for x = 1:1:m
          for y = 1:1:n
               if I(x,y) == i
                  g_val = [i x y];
                  pix_sort = cat(1,pix_sort,g_val);
               end
          end
     end
  dlmwrite('c:\zebraedges\pix_sort.txt', pix_sort,'-append','delimiter',
' ','newline','pc');
end

Open the file pix_sort.txt and read the numeric data into a matrix M
M=dlmread('c:\zebraedges\pix_sort.txt',' ');

Sort the rows in descending order using the first column which represents
the gray value.
M=sort(M,1,'descend');}
```

(b) label all pixels having the current gray value as belonging to the watershed or the catchment basin of a minima

(c) repeat the above steps until all pixels are processed

3. Find watersheds by taking the set complement of the basins or perform a binary operation that makes all watershed pixels have intensity 1 and others 0

3.6.3 Sorting Pixels

The pixels in the gray scaled image need to be sorted according to their gray values. To implement this in the Mathlab environment we scan the pixels in a breadth-first order and extract the pixels having the same gray value and append them to a matrix of three columns wherein the first represents the gray values, the second represents the x-coordinate and the third represents the y-coordinate of the extracted pixel. The code is provided in the table below.

3.6.4 Minima Detection

Minima in the image are all those points whose neighbohood pixels have equal or greater gray scale value than the point under consideration. If this pixel is not on the boundary then it will have 4 or 8 neighbors depending upon the connectivity used. But if it lies on the boundary then it can have 3 to 5 neighbors depending upon whether it lies on the corner or in the first or last row (or column). The algorithm should take care of these conditions. One way to get around this is to perform a padding of the boundary with zeros and run the algorithm from the row 2 to row last-1, and column 2 to column last-1. As

Table 3.2: Matlab Pseudo-Code for Sorting the Pixels for Immersion Watersheds

```
I = imread('c:\zebraedges\zebra','jpg');
I = rgb2gray(I);
[m n]= size(I);
Sort the pixel positions in terms of their gray values in ascending order
pixsort = [];
for i = 0:1:255
     for y = 1:1:n
         for x = 1:1:m
             if I(x,y) == i
                gval = [i x y];
                pixsort = cat(1,pixsort,gval);
             end
         end
     end
  dlmwrite('c:\zebraedges\pixsort.txt', pixsort,'-append','delimiter',
' ','newline','pc');
  pixsort = [];
end
Open the file pixsort.txt and read the numeric data into a matrix M
M=dlmread('c:\zebraedges\pixsort.txt',' ');
```

and when a minima is detected it is stored in a cell array structure which resizes itself to accomodate the new minima. The following table gives the details of such an algorithm.

Table 3.3: Psuedo-Code for Regional Minima Detection for a Given Gray Scale Value

```
pad I with zeroes all around
k=1;
padI = zeros(m,n);
for j=1:1:n
    for i=1:1:m
        padI(i+1,j+1) = I(i,j);
    end
end

Get all the minima and put them in a cell array Min
Extract all pixels with the same gray values and put them in a matrix Q
[r s]=size(Q);
Min=cell(1,256);
for t=1:1:r
    x=Q(t,2)+1;
    y=Q(t,3)+1;
        if (padI(x-1,y-1)>padI(x,y))&&(padI(x-1,y)>padI(x,y))
            if (padI(x-1,y+1)>padI(x,y))&&(padI(x,y+1)>padI(x,y))
                if (padI(x+1,y+1)>padI(x,y))&&(padI(x+1,y)>padI(x,y))
                    if (padI(x+1,y-1)>padI(x,y))&&(padI(x,y-1)>padI(x,y))
                        Min{1,k}=[padI(x,y) x y];
                        k=k+1;
                    end
                end
            end
        end
end
```

3.6.5 Variants of the Watershed Method

Two variants of implementing the Watershed Transformation are discussed in this section. The first one uses the Toboggan, and the second one uses the Triangulated Terrain simulation.

The Modification of the Toboggan Simulation

The Toboggan simulation can be modified to perform simultaneous flooding of the pixels identified as minima. The following is the algorithm for implementing the improved Toboggan Segmentation.

- Scan image in row order
- if pixel is not already labelled
  - find its lowest gradient neighbor
  - slide to a labelled pixel or local minimum
  - update number of regions corresponding to identified minimas by flooding them simultaneously
  - find its lowest gradient neighbor
- output the labelled regions and determine the watershed lines

It is worth mentioning that this method does not account for plateaus that could occur in the topography of an image. This requires an separate treatment in the form of some modification of the algorithm. One way is to smooth/blur these plateaus using a *gaussian* filter.

The Triangulated Terrain Simulation

To represent watersheds using the rainfalling analogy, one needs to trace paths of steepest descent on the landscape. In particular, we are looking for terrain features like the ridges, valleys, and saddle points; for which information about the edges and their connections to one another need to be derived from the pixel values. One such method is deriving the Triangulated Irregular Network, known as TIN. It is a piecewise-planar approximation to a terrain where each facet of the approximation is a triangle. This is how [24] describes representation of the digital terrain. Advantages of such a representation are that it is a good approximation of the drainage characteristics of the landscape, and water always follows the path of steepest descent. In other words, every point on the landscape has a unique path of steepest descent.

For a point p, let the trickle-path(p) be a unique path of steepest descent from point p on the terrain. The Watershed of a point q is $\{p|q\epsilon trickle - path(p)\}$, which are all the points whose paths of steepest descent include q.

A TIN edge can be one of the following three physical features in a terrain:

- It is a *valley* if the normals to the faces incident on the edge both point towards it. These valleys correspond to the rivers in the terrain.
- It is a *ridge* if the normals to the faces incident on the edges both point away from it. These ridges correspond to the watershed boundaries.
- It is a *pit* if it is a local minimum in the elevation of the terrain.

The trickle-paths can be reversed locally on the faces of TIN to behave as paths of steepest ascent known as *trace-ups*. A trace-up is stopped when it encounters a ridge or a saddle-point in the terrain. Sometimes a watershed boundary is not a TIN edge but lies on a TIN face. To identify such occurrences the behavior of a *height profile function* about a point can be studied. Let the height profile function $h_{\epsilon,p}(\theta) : [0, 2\pi) \to \mathbb{R}$ at a point p be a function of the angle θ that returns the elevation $z \rightarrow \mathbb{R}$ at which the cylinder $(x - px)^2 + (y - py)^2 = \epsilon^2$ intersects the terrain at an angle θ from p. A local maximum of the $h_{\epsilon,p}(\theta)$ refers to a direction where the flow of water splits around p, thereby, implying the presence of a ridge at p. On the other hand, its local minimum signifies a direction where the flow of water is such that it collects at p, thereby, implying existence of a valley at p. Performing an ϵ perturbation of p in the direction of each local maximum of $h_{\epsilon,p}(\theta)$ gives rise to multiple paths of steepest ascent for p. These help identify the family of ridges, valleys, and saddle points of the terrain or the landscape.

The drainage compelxity on a terrain allows for the following characterizations:

- If the local maxima and minima for $h_{\epsilon,p}(\theta)$ of every TIN point occur only at its ridge and valley edges, then it behaves as a nice terrain.
- If the local maxima and minima for $h_{\epsilon,p}(\theta)$ can occur on TIN faces and the trace-up paths along incoming ridges end-up in the interior of the ridges, then it behaves as a normal terrain.
- If $h_{\epsilon,p}(\theta)$ has no restrictions whatsoever, then the elevations at various points cannot be related to it, and the terrain then behaves as a nasty terrain.

The above differences in the terrain give rise to different watershed lines as explained in [24].

3.7 A Comparison

An interesting fact is that the two analogies do not generally give rise to the same watersheds, see 4.3. The reason being the watersheds depend on the support of the operation, i.e. the neighborhood over which the operations are performed. In the case of rainfalling analogy the neighborhood is the set of pixels that are along the path of the steepest descent of a pixel. On the other hand, in the case of flooding/immersion one needs the information about all pixels that belong to one threshold level prior to the current one. Thus, unless all these pixels are processed one cannot proceed to the next higher level. Hence, compared to the immersion analogy the rainfalling analogy has comparatively fewer pixels that need to be processed.

3.7.1 Comparison of Edge Detection and Watershed Boundary Detection Procedures

Applying heuristics before choosing the segmentation technique takes one step closer to getting desired results. Watershed Transform works on the whole image as it processes all pixels with the same intensity before moving-up one level while edge detection involves a filter matrix that processes a pixel at a time and then moves to the next.

The Watershed Transform utilizes the pixel intensity values to determine the network of steepest descent paths as in the case of rainfalling or proceeds from one threshold level to another in progression as in the case of immersion. In the latter case, the Watershed Transform can perceive the intervening changes at every threshold level. The edge detection methods proceed from one pixel to the next in a sequence and the use of the mask is on the immediate neighborhood of that pixel. The results of the edge detection can be only understood when all the pixels are completely processed. In a noiseless image, there is no loss of information upon the implementation of the Watershed Transform. But in practice, the Watershed Transform is applied on an image which is preprocessed, by taking say it normed gradient. In edge detection, the loss of information can easily occur at the bordering pixels where the mask goes out of bounds because of differing matrix dimensions and does not completely map the image pixels.

3.8 Problems that Affect Detection of Edges

Watershed Transform is a powerful method in the sense that it produces a complete division of the image into non-overlapping regions and does not need any kind edge-linking, unlike other edge detection methods. But it is marred by oversegmentation, sensitivity to noise, poor detection of significant regions with low contrast boundaries and thin structures [56, 58, 57].

3.8.1 Noise

A noise image, $g(x,y)$, can be considered as the original image, $f(x,y)$ corrupted by, say additive white gaussian noise, $\eta(x,y)$. This can be modeled by the following expression.

$$g(x,y) = f(x,y) + \eta(x,y) \tag{3.5}$$

[3, 56, 57] take a look at the noise probability density models in the spatial and the frequency domain, and use the statistics like mean, and variance to review methods of reducing the noise factor in the image.

The *space of noise level functions* can be defined as the variation of the standard deviation of noise with respect to image intensity, as given below.

$$\tau(I) = \sqrt{E[I_N - I]^2} \tag{3.6}$$

where I_N is the observation and $I{=}E(I_N)$.

A noise function is, thus, modeled by the mean noise level function, $\overline{\tau}$, and the eigenvectors, ${\eta_i}_{i=1}^m$, of the noise space.

$$\tau = \overline{\tau} + \sum_{i=1}^{m} \beta_i \eta_i \tag{3.7}$$

Here the coefficient β_i is assumed to have a gaussian distribution, i.e. $\beta_i \backsim \mathbb{N}(0, v_i)$, and the function τ positive everywhere.

Random noise in an image can be reduced by filtering with a median filter, SUSAN denoiser or other smoothing filters. Since smoothing operations tend to thicken this random noise into false edges, effects of noise can be reduced or eliminated only if these transitions are captured. For this purpose, some preprocessing like image averaging and image smoothing could be performed on the image.

Image Averaging

In this case an averaging is performed over a set of different noisy images representing the same image, and helps reduce the amount of noise in a summation of these images.

Image Smoothing

Smoothing the effects of the noise can be done by using a filter mask that acts on the immediate neighborhood of each pixel. These cause blurring and thereby, bridge the gaps in the contours and edges in the image which otherwise were open curves or lines. False edges can, thus, be blurred out to decrease their contrast with the true edges in an image.

3.8.2 Oversegmentation

Oversegmentation in an image produces too many regions or contours which obscure the relevant edges. This can be reduced by keeping fewer minima which can be done by markers. Another technique is that of a scale-space approach wherein regions of interest can be selected by using an appropriate nonlinear or morphological filter, [58]. Various preprocessing and postprocessing techniques also help reduce oversegmentation as can be seen in [59].

3.8.3 Significant Areas with Low Contrast Boundaries and Thin Structures

These can face poor detection if their signal to noise ratio is not too high at the contours. This can pose problems in, say, in finding contours between gray matter and bone. Reference [58] suggests use of active contour models called SNAKES.

3.9 History of Work

The earliest algorithms were based on operations that were local to each pixel and performed in an iterative manner. Beucher and Lantujoul suggested a method that expanded influence zones around local minima within their gray scale levels via binary thickenings until idempotence was achieved. Beucher later showed that a watershed can be calculated by performing gray scale thinning on the image until the edge reduces to one-pixel width. These iterative or sequential methods rely on scanning the pixels in a predefined order where the new value of each pixel is immediately taken into account in the processing of the subsequent pixels. Friedlander and Meyer proposed an algorithm based on horizontal scans.

Beucher also proposed a directed graph to represent a gray scale image. Each pixel is represented by a node in the graph connected to those neighboring pixels with strictly higher gray scale value. These directed connections are generated by the geodesic dilations of the pixels at successive gray levels. Iterative procedures are then used to identify the nodes which are the divide points, or the watershed pixels, see 4.2. These algorithms simulate the flooding of the landscape, propagating flood waters from the minima in height order, and building watershed lines when floods from different catchment regions meet.

3.10 Modifications to the Watershed Transform

Many existing watershed algorithms are improvements to the immersion algorithm, or the rainfalling algorithm, and sometimes are also a combination of both. The following is a list of instances where a certain technique brings about a desired modification or an improvement in the watershed contours.

- Applying Watershed Transform on the gray scaled gradient of the filtered image
- Marker-based minima detection to identify connected sets of pixels of a predetermined region in an image
- Identifying connected sets of pixels by similarity, reconstruction, labelling, regional maxima and minima, h-domes, etc

- Using efficient distance transforms that produce good segmentation

We investigate some of these methods in the next chapter.

Chapter 4

Improving the Watershed Transformation

4.1 Reducing Oversegmentation

The Watershed Transform, in either of its versions, suffers as we have seen from oversegmentation; noise and minor features generate watersheds which obscure the main contours present in the image. There are three methods which have been used to reduce oversegmentation: (i) preprocessing the image, (ii) modifying the watershed algorithm, and (iii) postprocessing the image.

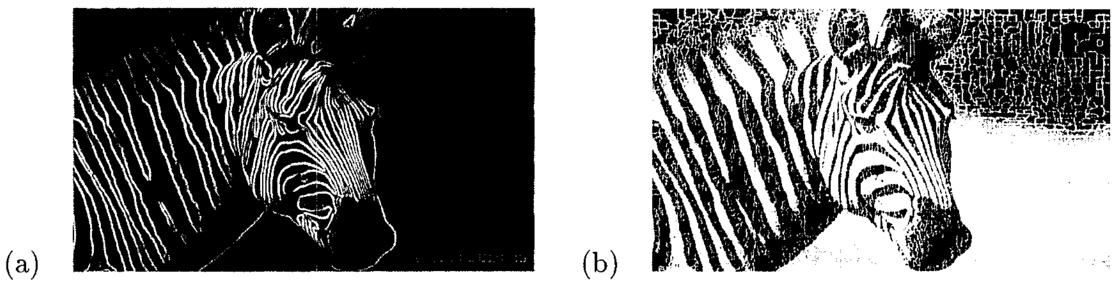


Figure 4.1: Watersheds (b) of the Zebra Obtained from the Gaussian Normed Gradient Image (a) with $\sigma = 1$. There is still enormous oversegmentation.

We have already considered preprocessing the image with a Gaussian to smooth it, thereby removing noise, and the use of the normed gradient filter to exaggerate the presence of sudden changes in intensity corresponding to boundaries or edges in the image. However even when this is done there can still be considerable oversegmentation as the images in figure 4.1 illustrate.

In the next two subsections we consider the use of *morphological methods* to both preprocess and postprocess the image. The last section outlines some of the ideas which have been incorporated into the watershed algorithms to reduce oversegmentation.

4.1.1 Morphological Methods

In biology, morphology is the study of the structure of living things. Mathematical morphology applied to images has the same connotation; that is it studies the structure of the image formed from its boundaries, edges and its components. For example, we could consider that convexity was an important property of a particular image and isolate the convex components. The natural language for gray scale morphology is set theory applied to $\mathbb{Z}^3 = \mathbb{Z}^2 \times \mathbb{Z}$ and it subsets, since a point of a gray scale image is a point in $\mathbb{Z}^2$ together with a pixel value in the $0 - 255$ range. Technically mathematical morphology is a collection of set theoretic operators defined on an infinite lattice. It was first investigated by Matheron [40] and Serra [47, 48].

There are four basic morphological operations on sets in $\mathbb{Z}^2$, *dilation*, *erosion*, *opening* and *closing*. Let $A, B \subset \mathbb{Z}^2$, and let $\overline{A}$ denote the complement of A. This will usually be with respect to some subset of $\mathbb{Z}^2$ since all our images are compactly supported. The

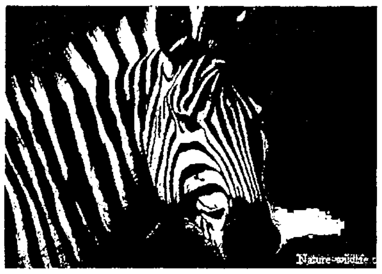


Figure 4.2: The Binary Zebra Image. In this section we shall use this to demonstrate the effects of the morphological operators.

translation of A by $b = (b_1, b_2) \in \mathbb{Z}^2$ is the set

$$A_b = \{m : m = a + b,\ a \in A\}$$

where $a + b = (a_1 + b_1, a_2 + b_2)$. The reflection $\hat{A}$ of A is the set

$$\hat{A} = \{m : m = -a,\ a \in A\}.$$

The *dilation* $A \oplus B$ of A by B, and the *erosion* $A \ominus B$, are the sets

$$A \oplus B = \{m : \hat{B}_m \cap A \neq \emptyset\} \qquad A \ominus B = \{m : B_m \subseteq A\}.$$

We don't need parentheses here because $\widehat{B_m}$ will denote the reflection of B_m. The relationships can be made more symmetrical by noticing that the definition for dilation is equivalent to $A \oplus B = \{m : \hat{B}_m \cap A \subseteq A\}$.

A morphological operator applied to an image requires the specification of one or more *structuring elements.* A structuring element is usually a regular geometric figure such as a square or rectangle, the size of which defines a *scale* for the morphological operation, cf. §4.1.2. For the dilation and erosion operators just defined, B plays the role of the structuring element for (the image) A. A morphological operator interprets the geometrical structure of the image in terms of the structuring elements. Figure 4.3 shows the effect of erosion and dilation on the zebra binary image Figure 4.2 using a single square 3×3 pixel structuring element.

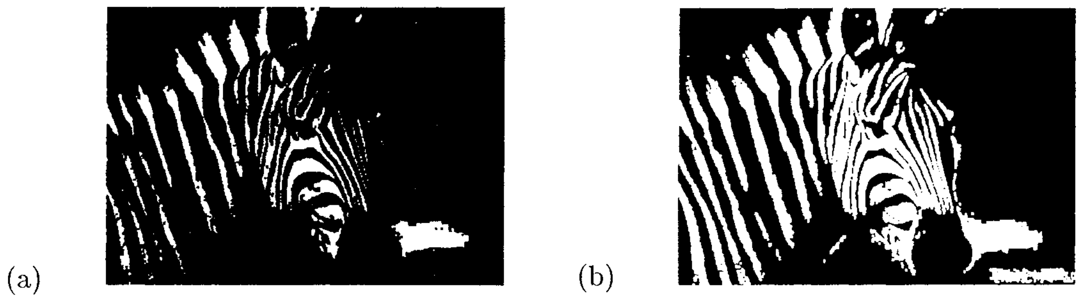

(a) (b)

Figure 4.3: The Effect of (a) Erosion and (b) Dilation on the Binary Zebra Image Using a Square 3×3 Pixel Structuring Element. The combined effect of the two operators is to filter out elements in the original image which do not comply with the scale of the structuring element.

In terms of the structuring element B and the set A, the *opening* $A \circ B$, and *closing* $A \bullet B$ operations are defined by

$$A \circ B = (A \ominus B) \oplus B \qquad A \bullet B = (A \oplus B) \ominus B.$$

The action of both the opening and closing operators is to smooth an image. However, whereas the opening operator tends to replace a thin section of a contour with a complete break, the closing operator will thicken any thin sections; it will also join up any small breaks in a contour. Figure 4.4 contrasts the actions of these two operators.

Let $I : D_I \rightarrow [0, 255] \subset \mathbb{Z}$ denote a gray scale image. Thus $I(p)$, $p \in D_I$ is the pixel value at p. The basic morphological operators can be extended to gray scale images. In fact as pointed out by Serra [47]-[51] any function $g : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$ which is *increasing*, that is

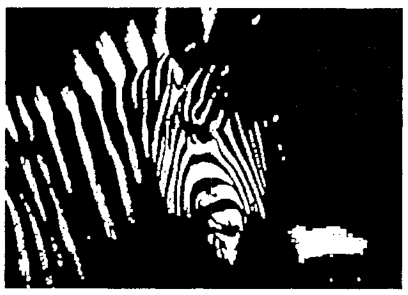
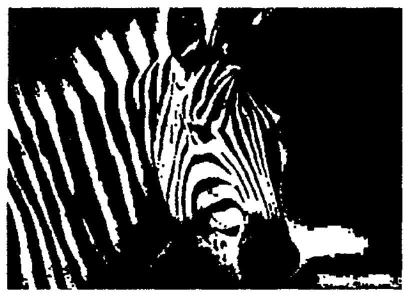
(a)  (b) 

Figure 4.4: The Effect of Opening and Closing on the Binary Zebra Image. Image (a) shows the effect of opening and image (b) the effect of closing. Notice that (b) has fewer breaks in the contours then either (a) or the original image. Whereas (a) has more breaks than the original image.

if $X \subseteq Y \subset \mathbb{Z}^2$ then $g(X) \subseteq g(Y)$ for all such subsets X, Y, can be extended to gray-scale images. This is because a gray scale image I has a *threshold decomposition*. This is the collection of nested sets

$$T_\ell = \{p \in D_I : I(p) \geq \ell\}$$

where $T_j \subseteq T_{j-1}$. The extension to gray-scale images is given, for all $p \in D_I$, by

$$g(I)(p) = \max\{j \in [0, 255] : p \in g(T_j(I))\}.$$

Each of the operators we have introduced have the required property. For example if we consider the dilation of I by a gray scale structuring element J, then $I \oplus J$ is defined by

$$(I \oplus J)(p) = \max\{j \in [0, 255] : p \in (I \oplus J)(T_j(I))\}.$$

In actual implementations the pixel values do not have to be in the range $0 - 255$, they can even be real valued, because of the action of image operators such as the Gaussian filter. It is easy to rewrite the definition of the gray scale dilation operator and the other gray scale morphological operators without this assumption:

$$(I \oplus J)(r, s) = \max\{I(r - x, s - y) + J(x, y) : (x, y) \in D_J,\ ((r - x), (s - y)) \in D_I\}$$

$$(I \ominus J)(r,s) = \max\{I(r+x, s+y) - J(x,y) : (x,y) \in D_J,\ ((r+x),(s+y)) \in D_I\}$$

$$I \circ J = (I \ominus J) \oplus J \qquad\qquad I \bullet J = (I \oplus J) \ominus J.$$

Figures 4.5 and 4.6 demonstrate the action of the gray scale morphological operators on the zebra image.

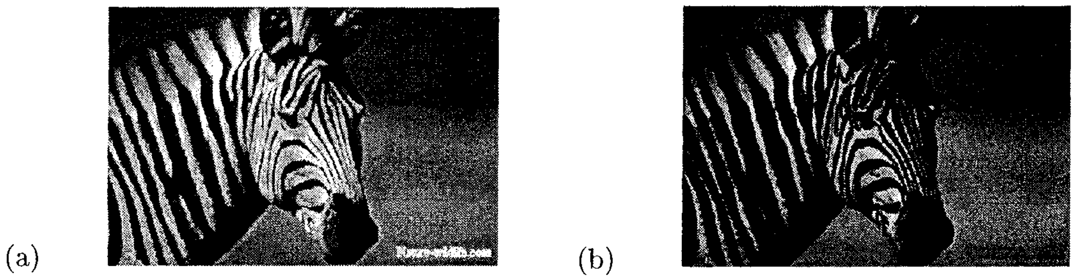


Figure 4.5: The Effect of the Gray Scale Morphoplogical Dilation (a) and Erosion (b) Operations Applied to the Gray Scale Zebra Image Using a Square 3 × 3 Pixel Structuring Element.

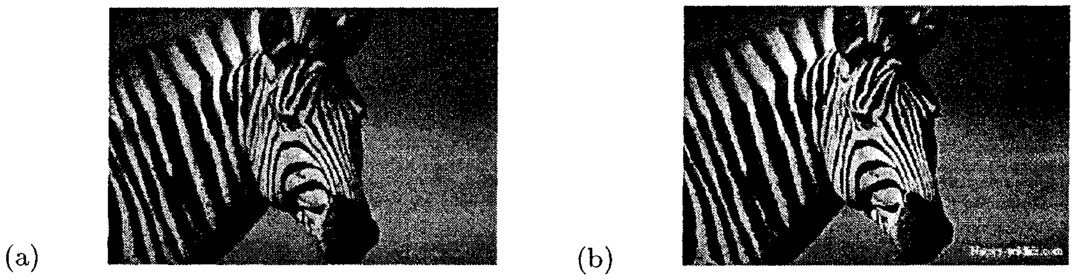


Figure 4.6: The Effect of the Gray Scale Morphoplogical Opening (a) and Closing (b) Operations Applied to the Zebra Image Using a Square 3 × 3 Structuring Element.

Morphological smoothing and gradient operators can be defined which have different properties from those derived from linear filters such as the Gaussian. Because they are nonlinear the transition regions in the transformed image usually have much sharper edges. For a morphological smoothing operator the features omitted from the smoothed image are those which cannot be approximated by the structural elements. The residual image, the difference between the original image and the smoothed image, can be treated to remove most of the noise and the purified residual can then be recombined with the smoothed image. We do not investigate the effect of this process in the thesis.

A morphological smoothing operation results from the action of an opening operator followed by a closing operator defined with respect to a set of structuring elements. For a single structuring element J the smoothed image $\mathcal{S}(I)$ depends upon the scale of the structuring element,

$$\mathcal{S}(I) = (I \circ J) \bullet J.$$

The proportion of high and low intensity pixels is reduced by the smoothing operation, Figure 4.7.

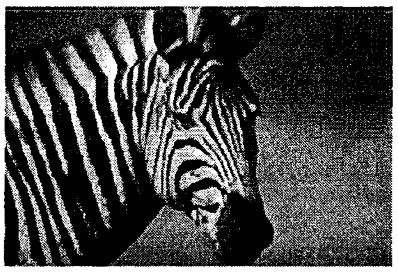
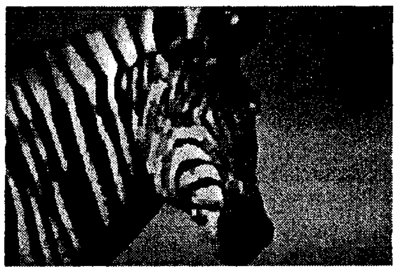
(a)  (b) 

Figure 4.7: The Effect of Gray Scale Morphoplogical Smoothing on the Zebra Image with (a) a Square 3×3 Pixel Structuring Element and (b) a Square 5×5 Pixel Structuring Element.

The *morphological gradient* $\mathcal{G}(I)$ of a gray scale image I with respect to a structuring element J is defined to be

$$\mathcal{G}(I) = (I \oplus J) - (I \ominus J).$$

Notice that the thickness of the edges depends upon the size of the structuring element.

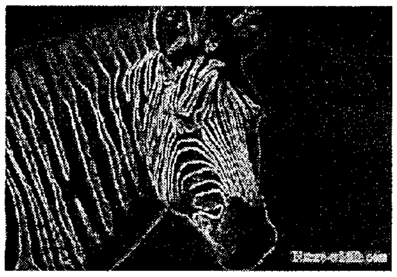
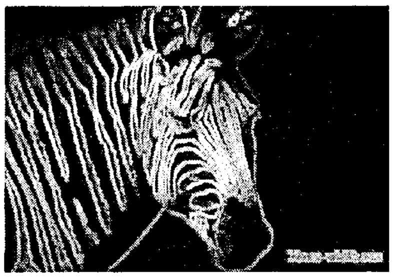
(a)  (b) 

Figure 4.8: The Effect of the Gray Scale Morphoplogical Gradient Using (a) a Square 3×3 Pixel Structuring Element and (b) a Square 5×5 Structuring Element.

Figure 4.8 illustrates the morphological gradient for the zebra image at two different scales. Normed gradient operators such as the Sobel or normed gradient Gaussian operator tend to introduce noise through *quantisation error*. That is they do not act naturally on integer valued pixels but give real values which have to be rounded up or down. The morphological operators do not introduce this sort of error.

4.1.2 Mutiple Scale Image Analysis

A gray scale image has a natural range of scales which run from the single pixel to the whole image. Features present in the image can be enhanced or sometimes removed altogether by a suitable choice of scale. A recent application of this idea uses wavelets to obtain a resolution of an image into scales associated with a wavelet basis. A variant of the Canny edge detector has been developed which detects edges at the different resolutions obtained in this way, [39].

The resolution of an image at different scales does not depend upon a wavelet basis and there has been considerable work on the scales defined by the standard deviation σ of the linear Gaussian filter. In fact the two concepts can be related through the *Gabor* *wavelet*, a Gaussian windowed sinusoidal wave, which in one dimension is defined by

$$g(x) = G(x) \exp(-\frac{x^2}{2\sigma^2}) \exp i\eta x$$

where $G(x) = \sigma^{-1/2}\pi^{-1/4}\exp(-x^2/2\sigma^2)$ is the Gaussian function.

The morphological operators have scales determined by the structuring element. The effect of varying this for the morphological gradient can be seen in Figure 4.8. One possible refinement of the morphological gradient is to remove lower order contributions by erosion. For example, if $\tilde{J}$ is a structuring element such that $||\tilde{J}|| < ||J||$ in some appropriate norm then

$$\mathcal{G}(I) = ((I \oplus J) - (I \ominus J)) \ominus \tilde{J}$$

is a morphological gradient from which the contribution from the structuring element $\tilde{J}$ has been eroded.

4.1.3 Geodesic Reconstruction

Even with the use of the multiscale morphological gradient the watershed algorithm still gives considerable oversegmentation. To overcome this a further preprocessing step called *gray scale reconstruction*, [52, 53], is applied to the gradient image.

(a) (b)

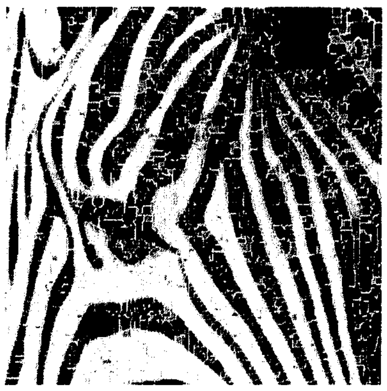
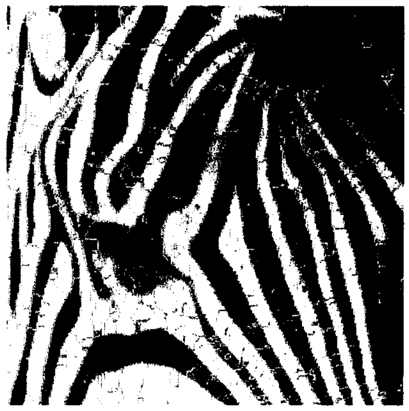
(c)  (d) 

Figure 4.9: The Watershed Transformation Applied to the Morphoplogical Gradient on the Zebra Image. Figure (a), still gives considerable oversegmentation. Figure (b) shows the result of the watershed transformation after applying gray scale reconstruction. Figures (c) and (d) are closeup sections of (a) and (b) respectively.

Gray scale reconstruction of an image involves the use of *markers* which are themselves (usually disconnected) subsets of the image. For binary images the process identifies the connected components of an image I which contain one or more pixels of a marker J. Thus the reconstruction $\rho_I(J)$ of I defined by J is given by

$$\rho_I(J) = \cup_{J \cap I_k \neq \emptyset} I_k$$

where I_k, $I = \cup_k I_k$, is a connected component of I.

We need an alternative definition which enables the connected components to be calculated from a given marker if this is to be of any use practically. For digital images a pixel either has four or eight neighbors, if we allow the pixels which are offset diagonally. Thus we can define one of two natural metrics on a gray scale image depending upon the connectivity. Let $d_I(x, y)$ denote the distance between the pixels $x, y \in D_I$, using either of the two topologies. The *geodesic dilation of size n* is defined to be

$$\delta_I^{(n)}(J) = \{p \in I : d_I(p, J) \leq n\}.$$

This can be obtained by iterating the elementary dilation $n = 1$, n times from which we deduce that $\delta_I^{(j)}(J) \subset \delta_I^{(j+1)}(J)$, $j = 0, 1, .., $ where $\delta_I^{(0)}(J) = J$.

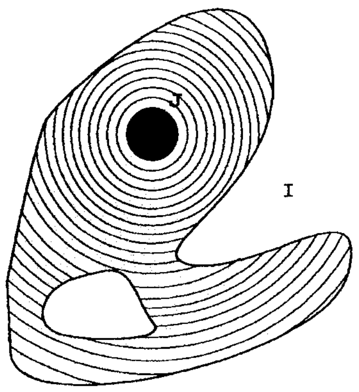


Figure 4.10: The Geodesic Dilation of the Marker J Inside a Connected Set $I \subset \mathbb{R}^k$ Using the Euclidean Norm.

The relation between geodesic dilation and reconstruction is clearly seen from Figure 4.10. The reconstruction, that is the set of connected components of image I with respect to the marker J, is obtained by iterating elementary dilations until stability is obtained

$$\rho_I(J) = \cup_{j \geq 1} \delta_I^{(j)}(J).$$

The gray scale version of this binary reconstruction can be derived through the threshold decomposition which we introduced in §4.1.1. Let I and J be gray scale images with pixel values in $\{0, ..., 255\}$. Define $J \leq I$ to mean that $J(p) \leq I(p)$ at every point $p \in D_I$. Then the set theoretic definition of the gray scale reconstruction $\rho_I(J)$ of I by J is given for every $p \in D_I$ by

$$\rho_I(J)(p) = \max\{j \in [0,255] : p \in \rho_{T_j(I)}(T_j(J))\}.$$

The gray scale reconstruction extracts the peaks of the image which are labelled by the marker. The diagram Figure 4.11 shows how this would work for continuous marker function. Notice that essentially the extraction process drowns the neighborhood of a maximum of the marker which is contained in the neighborhood of a maximum of the image.

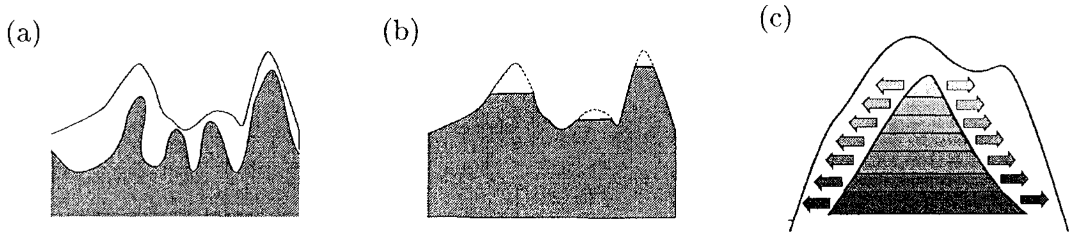


Figure 4.11: Gray Scale Reconstruction Using the Definition in the Text. Figure (b) results from gray scale reconstruction of the image in Figure (a) by the marker J (the gray filled contour). Figure (c) illustrates the *flooding* which occurs at each distinct gray level in the reconstruction process.

Gray scale reconstruction can also be defined in terms of elementary geodesic dilations which themselves are related to gray scale dilations. For this we need to introduce the *minimum* $\wedge$, and *maximum* $\vee$, operators which generate images $I \wedge J$ and $I \vee J$ respectively. They are defined at $p \in D_I$ by

$$I \wedge J(p) = \min\{I(p), J(p)\} \qquad I \vee J(p) = \max\{I(p), J(p)\}.$$

The elementary geodesic dilation $\delta_I^{(1)}(J)$ in the gray-scale case, $J \leq I$, is then given by

$$\delta_I^{(1)}(J) = (J \oplus B) \wedge I$$

where B is some gray scale structuring element. Thus the *gray scale reconstruction* of I from J is obtained as a limit of elementary gray scale dilations,

$$\rho_I(J) = \vee_{n \geq 1} \delta_I^{(n)}(J)$$

where $\delta_I^{(n)}$ is the dilation obtained by iterating $\delta_I^{(1)}$ n times. A dual formulation of gray scale reconstruction can be given in terms of geodesic erosions [52]. Thus the dual gray scale reconstruction $\rho_I^*(J)$ of the image I from the marker J, $I \leq J$ is given by

$$\rho_I^*(J) = \wedge_{n \geq 1} \epsilon_I^{(n)}(J)$$

where the elementary geodesic erosion $\epsilon_I^{(1)}$ is given by

$$\epsilon_I^{(1)}(J) = (J \ominus B) \vee I.$$

A *regional maximum* of a gray scale image is a connected component of pixels which has a given value h such that all neighboring pixels have a lower value. Thus if we use $I - 1$ as the marker then $M(I) = I - \rho_I(I - 1)$ is an image consisting of the regional maxima. This definition is easily modified to result in an image consisting of the regional maxima of a given height or pixel value. Another useful idea for image segmentation is the *h-dome* which is a connected set of pixels in the image such that the difference between the maximal and minimal pixels in the set is strictly less than h. Pixels in the neighborhood of the h-dome have values stricly less than the minimal h-dome value. The h-dome image $D_h(I)$ of the h-domes is defined by

$$D_h(I) = I - \rho_I(I - h).$$

4.2 Modifying the Watershed Transformation

Gray scale reconstruction preprocessing can considerably reduce the amount of oversegmentation which results from the watershed transformation. In this section though we consider techniques which have been used to directly attack the oversegmentation problem. Since the watershed transformation is applied to a gradient image, we have to consider methods which either (i) amend the gradient image so that local minima only correspond to the objects of interest in the image, or (ii) eliminate the noisy or irrelevant contours so that only contours of the target objects remain.

The first approach uses *markers* to locate the objects to be detected and also their backgrounds, [49, 50], [41]. The method is dependent upon a way of identifying suitable markers in the image.

The alternative approach aims at flooding neighboring catchment basins according to some specified rules. The methods that have been developed for this are collectively called *region growing algorithms*, [42], [50], [43], [36].

4.2.1 Markers

Once background and foreground markers have been obtained for an image the gradient image is modified so that pixels belonging to the markers are given the value 0 and all unmarked catchment areas are flooded. The watershed transformation is then applied to the modified gradient image.

The main difficulty in this approach is how to define suitable markers for the image.

If the image has objects which have well defined h-domes, such as light coloured particles against a dark background, the problem is fairly straightforward since the h-domes can be extracted and then thresholded to yield the markers. The zebra image is harder to handle because the stripes are "feathered" along the neck. Therefore, we apply an opening-by-reconstruction operation followed by a closing-by-reconstruction operation which removes many of the extreme local variations from the image.

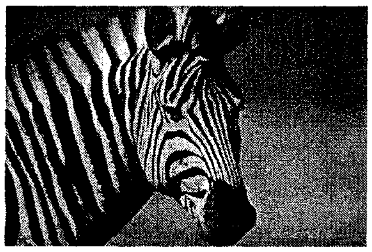
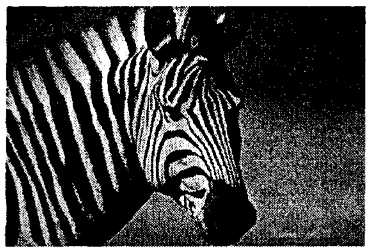
(a)  (b) 

Figure 4.12: Gray Scale Reconstruction by Opening and Closing. In (a) the eroded zebra image is used as the marker in the reconstruction. Figure (b) is the gray scale reconstruction of image (a) using the marker obtained from it by dilation.

Figure 4.12(a) illustrates opening-by-reconstruction applied to the zebra image. The marker in this case is the image resulting from the erosion of the zebra image using a 3×3 square pixel structuring element. Figure 4.12(b) is the result of closing-by-reconstruction applied to 4.12(a). In this case the marker is obtained from dilation applied to (a) using a 3×3 square pixel structuring element.

Now that most of the trivial detail has been eliminated from the image we can try to locate the regional maxima using the method of subsection 4.1.3. This should locate most of the stripes of the zebra since it can be seen from Figure 4.12 that most of them contain regional maxima. Unfortunately, the unstructured background also has a lot of maxima and so we are unlikely to get a good segmentation this way. Figure 4.13 shows the result

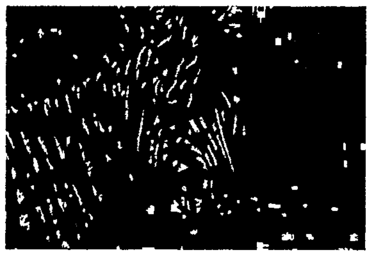
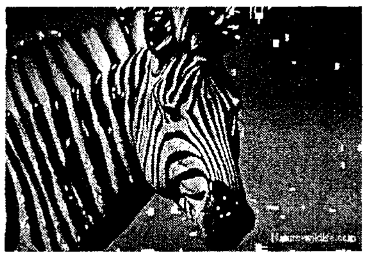
(a)  (b) 

Figure 4.13: The Markers for the Zebra Image. Figure (b) shows the markers in (a) superimposed on the zebra image.

of this process. This figure seems to be the best that we can do. A smaller structuring element gives too many maxima, whereas larger structuring elements give to few.

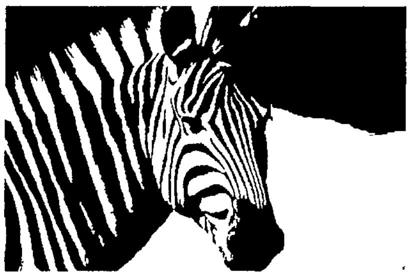
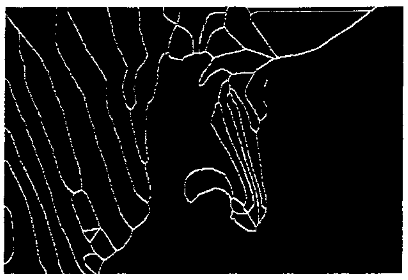
(a)  (b) 

Figure 4.14: The Background Markers for the Zebra Image. Figure (a) is the thresholded image used to obtain the background markers in (b).

The markers for the background suffer from similar problems. Thresholding the image should give mainly the background pixels but Figure 4.14(a) shows that the background cannot be entirely selected using this approach. Figure 4.14(b) shows the crest lines that are going to be used as the background markers. They are in fact the SKIZ (skeleton by influence zones) of the binary image in (a). This can be obtained by first computing the distance transform of (a) and then using the watershed transform to get the crest lines. The markers are then incorporated into the gradient image so that they are the regional minima. The watersheds obtained by this process are displayed in Figure 4.15(a). Figure

4.15(b) shows the superposition of the watersheds on the zebra image.

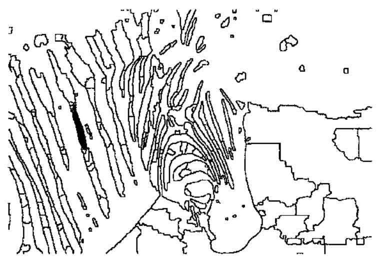
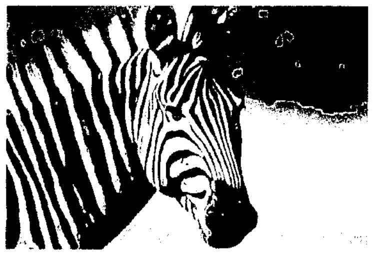
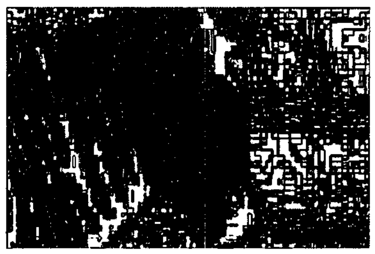
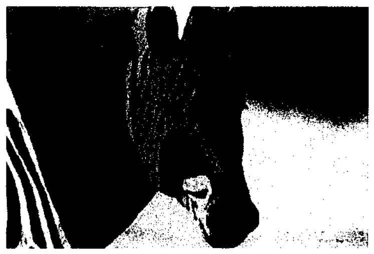
(a)  (b) 

Figure 4.15: The Watersheds for the Zebra Image Obtained from Markers. In Figure (b) the watersheds in (a) are superimposed on the zebra image.

4.2.2 Region Growing

This method is particularly useful if it is very difficult to derive suitable markers from an image. It also raises the possibility of a technique which can be applied automatically without the detailed user interaction necessary for the generation of markers. The approach starts from the gradient image and aims at the removal of the irrelevant contours from the gradient image watersheds. A minimum ℓ in the normed gradient image is a set of

(a)  (b) 

pixels contained in some open ball of I which have the same gray scale value within some given tolerance (remember the normed gradient produces real gray scale values). Each minimum is contained in a catchment basin B_ℓ which is bounded by a curve along which the normed gradient is a maximum. Within the region R_ℓ so defined a common gray scale

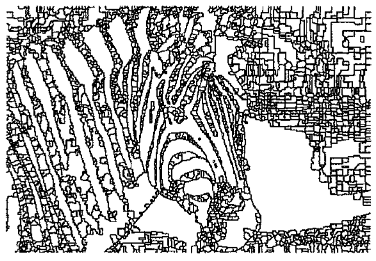
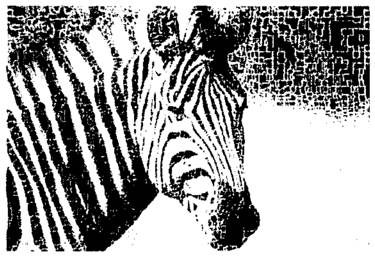
(c)  (d) 

Figure 4.16: The (a) Minima and (b) Mosaic Image Associated with the Zebra Image. Figure (c) are the watersheds obtained from the morphological gradient using a 3×3 square structuring element. Figure (d) shows the watersheds imposed on the zebra image.

value is assigned to all the pixels. This is chosen so that it is related to the gray values of the original image. For example, the value could be chosen to be the average gray level of the original image within the bounded region. A common choice for the gray scale value in C_ℓ is $\inf\{I(p) : p \in \ell\}$. If the morphological gradient is used instead then the pixels belonging to ℓ have the same gray scale value.

This process is repeated for every minimum in I. The resulting image $I^{(1)}$ is called the *mosaic image*. This is an image with a simplified set of gray values, see Figure 4.16. The process can be iteratively repeated. The mosaic image $I^{(2)}$ for example is obtained by deriving the normed gradient image of $I^{(1)}$ and deriving its mosaic image.

The iterative generation of mosaic images has a graph theoretic interpretation. In the literature this idea first appears in Rosenfeld's papers [45, 46]. Recent work involves the concept of *topological watersheds*. Essentially, the idea is to retain the connectivity of the image throughout the process described above. Thus if two pixels of the original image are separated by a crest, they remain separated by a crest of the same height in the mosaic. The condition that this should be the case is that the mosaic arises through a *topological thinning*, [43], [35], [36].

4.3 Rainfalling and Immersion: A Comparison

There are many variants of the the two basic watershed transformation algorithms. Most of them incorporate a version of region growing as part of their processing. In this section we discuss two variants of the transformations which were introduced in Chapter 3.

4.3.1 A Modified Immersion Watershed Algorithm

The sorting step divides the domain of the normed gradient image $\mathcal{G} = \mathcal{G}(I)$, (or the morphological gradient image) $D = D_{\mathcal{G}(\mathcal{I})}$ into disjoint level sets

$$D_h = \{p \in D : \mathcal{G}(p) = h\}.$$

It is convenient, as usual, to consider that the gray scale values of all images belong to $[0, 255] \subset \mathbb{Z}$. Starting at level 0 catchment basins are each uniquely labelled at the altitude at which they first appear. At level h the catchment basins detected to level $h-1$ will therefore all have been assigned labels. The following are the new possible connected components which can arise in the level h classification.

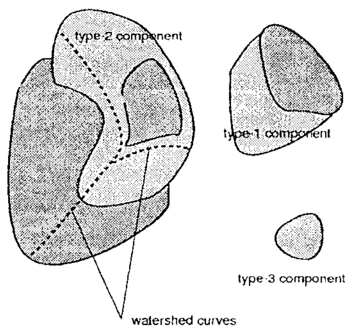
Type-1 component: A connected level h component which is not connected to any component of level less than h.

Type-2 component: A connected level h component which is connected to precisely one catchment basin of altitude less than h.

Type-3 component: A connected level h component which is connected to more that one catchment basin of altitude less than h.

When flooding raises the level of the water from level $h-1$ to level h, we see that each of these components has the following interpretation. A type-1 component is a new minimum at this level and therefore defines a new catchment basin. It must be assigned a new label. A type-2 component is flooded by the single catchment basin to which it is connected and so acquires the same label as this basin. Finally, for a component of type-3, the water rising from the several catchment basins to which it is connected will meet along watershed curves.

The standard implementation of the immersion watershed algorithm recognises three classes of pixels. The relation between them and the types of component at level h is also depicted in Figure 4.17.

(a)


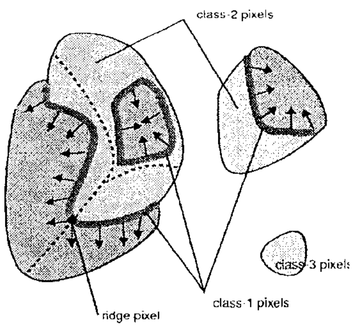
(b)


Figure 4.17: The Relation Between the Connected Components of Level h. Figure (a) illustrates the three different type of component which can occur. Figure (b) shows the relation between the pixel and component classification.

Class-1 pixel: Let $h_{\min}(p) = \min\{\mathcal{G}(q) : d_{\mathcal{G}}(q,p) = 1\}$. Then $p \in D_h$ is in class-1 if $\mathcal{G}(p) > h_{\min}(p)$. They for example lie on the descending edges of plateaus.

Class-2 pixel: A pixel $p \in D_h$ belongs to this class if it is contained in either a type-1 or type-2 component, but is not of class-1. Class-2 pixels for example lie in the interior of extended plateaus.

Class-3 pixel: A pixel $p \in D_h$ that belongs to a type-0 component is in this class. A pixel in class-3 has to be assigned a new label.

A possible ambiguity arises in the Vincent and Soeille algorithm [50], when flooding pixels in D_h. This occurs when the treatment of some class-1 pixels is considered. If the lower neighbors of the pixel have the same label then this pixel is given the same label. However, it is possible that the lower neighbors have different labels. To avoid introducing a bias into the segmentation process the authors of [38] introduce a new label *ridge* which is used to give the pixel a unique value. The modified algorithm is then *order-invariant* as the watersheds do not depend upon the order in which neighboring pixels were processed.

4.3.2 A Modified Rainfalling Algorithm

The rainfalling algorithm can also be modified to take care of ambiguities which arise in the labelling of class-1 pixels. The pseudocode for the basic rainfalling algorithm, which is called tobogganing in [38], is given as follows.

```
Step 1     Simulation of sliding: Record all the steepest descent directions
           for all class-1 and class-2 pixels in $D$.
Step 1.1   Simulate steepest descent sliding for each of the class-1
           pixels in $D$ by storing the lowest neighbors in a list. Push
           onto a FIFO queue as the seeds for region growing in step 1.2
Step 1.2   Simulate keep-sliding for all class-2 pixels in $D$ by region
           growing from class-1 pixels (using the FIFO queue in Step 1.2).
```

Step 2 Label all the class-3 pixels since they are the bottoms of new catchment basins.

Step 3: Steepest descent step: Assign a label to the unlabelled pixels (class-1,-2), by steepest descent (tobogganing) and then backtracking using a depth first search.

This algorithm too can be made order invariant by introducing the ridge label for the ambiguous pixels in Step 3. The authors of [38] claim that the two invariant watershed algorithms for immersion and rainfalling produce the same watersheds. However, we have noticed a subtle, but distinct difference in the watersheds for the zebra image produced by the two methods. The introduction of the ridge label which renders the algorithms invariant also seems to produce more plateaus in the watersheds than the variant versions.

4.4 Summary

The discussion of the Watershed Transformation which has been given in the thesis only touched on many of the recent developments, such as topological watersheds [36]. There are also innovations which lie beyond the scope of this thesis, such as *watersnakes* which derive the watersheds as the minima of an energy function defined in terms of the topographical distance function, [44].

The approach using markers to obtain the desired segmentation does not seem to work very well on images for which the background and foreground of an object is not clearly delineated. It is possible that the topological watersheds may lead to an effective method for handling these cases, but at the moment the region growing algorithms do not handle this situation very well either.

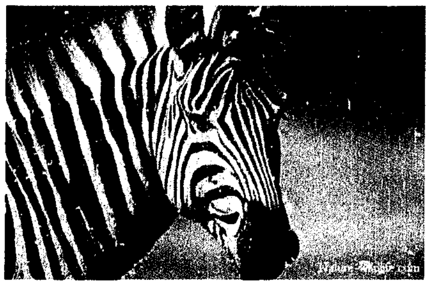
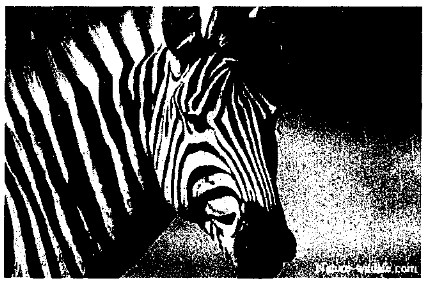
(a)  (b) 

Figure 4.18: The Segmentation of the Zebra Using the Modified Multiscales Gradient (4.1). Figure (a) $h = 10$, and (b) $h = 50$.

It might appear that deriving markers from the normed or morphological gradient should lead to an improvement in the generation of markers. However, we were unable to get any significant improvement this way. An approach which uses a threshold combined with several morphological scales produced much better results for the zebra image. This form of the gradient was introduced by Wang, [54]. It is given in terms of the operator,

$$\mathcal{G}(I) = \frac{1}{3}\sum_{i=1}^{3}[(I \oplus J_i - I \ominus J_i) \ominus J_i]$$

where (J_i) is a square $(2i+1) \times (2i+1)$ structuring element. Thus it is an average over the morphological gradients at three different scales with the contribution from a finer scale eroded. Small local minima are eliminated by dilating with a 2×2 structuring element B. The final form of the operator is

$$\tilde{\mathcal{G}}(I) = \mathcal{G}(I) \oplus B + h. \tag{4.1}$$

The constant h controls the number of segmentation regions. The larger h the fewer segmentation regions. Figures 4.18, and Figure 4.19 show some of the results obtained in this way.

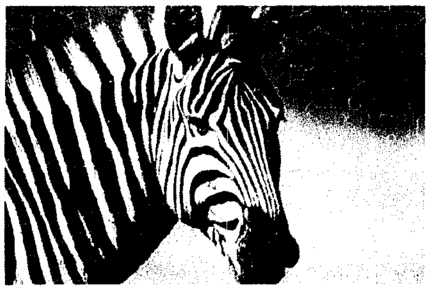
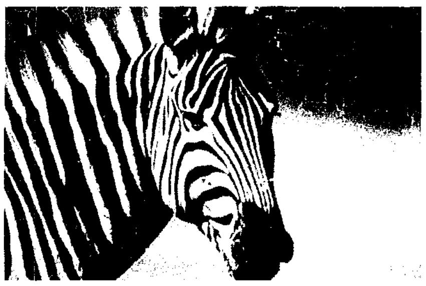
(a)  (b) 

Figure 4.19: The Segmentation of the Zebra Using the Modified Multiscales Gradient (4.1). Figure (a) $h = 110$, and (b) $h = 250$.

Chapter 5

Watershed Transform and Non-Morphological Processing

In addition to the methods discussed in chapter 4, there are other ways of analyzing the watersheds obtained in the Zebra image. One way is to resolve the watershed image in a wavelet basis and then look at each resolution or level information in the whole band/spectrum of the images in the basis. The theoretical basis of such a proposition is presented in the beginning of this chapter. But because of the time constraint we plan to experiment with it later. One other way can be to have different preprocessed treatments of the Zebra image with a smoothing operator and then apply the watershed transform on the images, thus, obtained. We tried this by using a Gaussian and a Laplacian operator with varying values of sigma or the standard deviation. The results of this experiment have been documented and analyzed in this chapter. Postprocessing treatments of the watersheds also reveal interesting features in the Zebra image. For example, the Canny operator can be applied on the Watershed Transform of the Zebra. This helps to visualize the link between the closed contours of the watersheds with the preservation of the true edges. One is forced to think of such a relation because, eventhough, Watershed Transform has a firm theoretical basis its implementation is marred with problems. One reason for such a school of thought is that the results obtained from the rainfalling method are different from those obtained by the immersion method. It is for this reason that one is tempted to prefer the Canny operator against the Watershed Transform. The experimentation with various pre- and postprocessing techniques is performed to shed more light on how the watersheds behave with respect to a variety of factors or operators.

5.1 Multiresolution

Pixel is the smallest element that the display or print hardware and software can manipulate to create letters, numbers, or graphics. Resolution determines the amount of information that appears on the screen, measured in pixels. A low resolution of, say, 640×480 makes items on screen appear large, although the screen area is small; while a high resolution of, say, 1024×768 makes overall screen area large, although individual items appear small. Multiresolution can then be thought of as representation and analysis of signals and images at more than one resolution. Performing a multiresolution analysis (MRA) helps identify features at a resolution that might go undetected at other levels or scales of resolution. [15, 13] provide a detailed treatment of this concept.

5.1.1 Multiresolution and Image Compression

MRA requires the use of a scaling function to create a series of approximations of a function or image, each differing by a factor of 2 from its nearest neighboring approximations. To create a sequence or a link between these neighboring approximations additional functions are needed that allow storage or encoding of the difference in information between adjacent approximations. These additional functions can be fourier series approximation, or wavelets. Wavelets are popular because both frequency and time scales can be associated with it and images can be compressed by mapping them with wavelets of varying frequency and limited duration. This stage called the image compression, therefore, contains all the information regarding the image. In contrast, the fourier series use sinusoids for approximations, and only the information pertaining to the frequency can be transmitted well, while that regarding the time is lost [13].

5.1.2 Multiresolution and Watershed Transform

A practical problem in realizing the goals of Watershed Transform is that of oversegmentation due to noise. When a boundary detection procedure, like the Watershed Transform, is applied on an image such an oversegmentation results in the appearance of unwanted edges in the watersheds. Applying Watershed Transform at different scales and then retrieving the one which has contours closest to the original image can help minimize oversegmentation and maximize the probability of getting optimal results. One is looking for the result whose intersection set with the original image has the maximum number of elements. This is the motivation for using the multiresolution analysis in close conjuction with the Watershed Transform.

The inherent properties of an image are its contrast, brightness, sharpness, and the color resolution. To detect boundaries in a high resolution image one can use either edge detection or the watershed transform. But to resolve a low resolution image for its boundaries watershed transform provides appreciable results when applied at high resolutions.

5.1.3 The Mathematical Foundation

The concept of multiresolution analysis is based on sequences V_n of subspaces which 'live' on a uniform structure generated by shift-invariance principles and dyadic dilation. These auxiliary spaces are spans of scaling functions of levels less than or equal to n such that a discretization can be solved efficiently, say, by preconditioned iterative methods, with the precondition inherited from the generating system, or frame, consisting of scaling functions.

The following definitions help characterize and theorize the concept of Multiresolution Analysis. These follow from [32], [33], [35].

Definition: An Orthonormal Wavelet is a function $\psi \in \mathbb{L}^2(\Re)$ such that the doubly indexed set $\{2^{j/2}\psi(2^j t - k)\}_{j,k \in Z}$ is an orthonormal basis of $\mathbb{L}^2(\Re)$.

Definition: A Multiresolution analysis (MRA) is an increasing sequence of subspaces $\{V_n\} \subset \mathbb{L}^2(\Re)$ defined for $n \in Z$ with

$$\ldots V_{-1} \subset V_0 \subset V_2 \subset \ldots\ldots$$

together with a function $\phi \in \mathbb{L}^2(\Re)$ such that

1. $\bigcup_{n=-\infty}^{\infty} V_n$ is dense in $L_2(\Re)$, $\bigcap_{n=-\infty}^{\infty} V_n = \{0\}$

2. $f \in V_n$ if and only if $f(2^{-n}) \in V_0$

3. $\{\phi(x-k)\}_{k \in Z}$ is an orthonormal basis of V_0

ϕ is called the scaling function of the MRA. V_0 is uniquely defined by ϕ through (3), and V_n is uniquely determined by (2). Also, ϕ need not be unique and a given family $\{V_n\}$ may have several different choices of ϕ.

5.2 Multiscale Analysis

The core of image processing and analysis techniques utilizes the image edge information stored in its gradient. By finding a local measure for the contrast of an image at different scales, it is less likely that the existing or the inherent noise will be missed. Thus, applying multiscale techniques to images helps describe and, thereby, extract the actual image edges. The use of wavelet transform to achieve compression, noise-reduction, enhancement, classification, and segmentation of gray scale images can be seen in [13, 17].

5.2.1 Edge Representation

When an image is read-in, the first thing to be done is to break-up the related information about the pixels and store it in a certain format. This constitutes image decomposition. The method chosen for image decomposition plays a crucial role in providing pixel level information which helps describe the edge pixels.

Let $I(x, y)$ be an image with components $I_n(x, y), n{=}1, \ldots, N$. Thus, at a given point the value of I is an N-dimensional vector. The gradient of I is given by the following differential [31].

$$dI = \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy$$

Its squared norm can be found as below

$$\begin{aligned}(dI)^2 &= \left(\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy\right)^2 \\ &= \left(\frac{\partial I}{\partial x}\right)^2 (dx)^2 + \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}dxdy + \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}dxdy + \left(\frac{\partial I}{\partial y}\right)^2 (dy)^2 \end{aligned} \tag{5.1}$$

$$\begin{aligned}(dI)^2 &= \begin{pmatrix} \left(\left(\frac{\partial I}{\partial x}\right)^2 dx + \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}dy & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}dx + \left(\frac{\partial I}{\partial y}\right)^2 dy\right) \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} \\ &= \begin{pmatrix} dx & dy \end{pmatrix} \begin{pmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} \end{aligned} \tag{5.2}$$

Its squared norm is given by

$$\begin{aligned}(dI)^2 &= \begin{pmatrix} dx \\ dy \end{pmatrix}^T \begin{pmatrix} \|\frac{\partial I}{\partial x}\|^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \|(\frac{\partial I}{\partial y}\|^2 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} \\ &= \begin{pmatrix} dx \\ dy \end{pmatrix}^T \begin{pmatrix} \sum\left(\frac{\partial I_n}{\partial x}\right)^2 & \sum\frac{\partial I_n}{\partial x}\frac{\partial I_n}{\partial y} \\ \sum\frac{\partial I_n}{\partial x}\frac{\partial I_n}{\partial y} & \sum\left(\frac{\partial I_n}{\partial y}\right)^2 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} \end{aligned} \tag{5.3}$$

It is this quadratic form that is called the first fundamental form. In [15], Scheunders proposes that the multimodal edge information is reflected by the eigenvectors of the 2×2 matrix in the first fundamental form. These eigenvectors denote the maximal and the minimal change. The eigenvalues of the above equation describe the rates of change in a multimodel image. For a gray level image (N=1), the largest eigenvalue is the squared gradient magnitude itself as given below, while the other eigenvalue is zero.

$$\lambda_1 = \|\nabla I\|^2$$

The corresponding eigenvector lies in the direction of the maximal gradient. Thus, the underlying idea is that if $\lambda_1 \neq \lambda_2$, then the eigenvectors can be uniquely oriented in simply connected regions. If $\lambda_1 \gg \lambda_2$, then the gradients of all bands are more or less in the same direction; while if $\lambda_1 \approx \lambda_2$ then there is no preferred direction.

5.2.2 The Dyadic Wavelet Transform

The non-orthogonal discrete wavelet frames introduced by Mallet are useful in describing the wavelet transform. Let $\theta(x,y)$ be a $2-D$ smoothing function. Suppose θ is differentiable, then we can define

$$\psi^1(x,y) = \frac{\partial\theta(x,y)}{\partial x}$$

and

$$\psi^2(x,y) = \frac{\partial\theta(x,y)}{\partial y}.$$

The Wavelet Transform of a gray level image $I(x,y)$ is then given by

$$D_s^1(x,y) = I * \psi_s^1(x,y)$$

and

$$D_s^2(x,y) = I * \psi_s^2(x,y)$$

where * denotes the convolution operator and

$$\psi_s^1(x,y) = \frac{1}{s^2}\psi^1\left(\frac{x}{s},\frac{y}{s}\right)$$

and

$$\psi_s^2(x, y) = \frac{1}{s^2}\psi^2\left(\frac{x}{s}, \frac{y}{s}\right)$$

denote the dilations of the functions ψ^i, and $s = 2^j$, $j = 1, \dots, d$ is the scale parameter. This d is the depth of the dyadic wavelet transform. $D_{2^j}^1$ and $D_{2^j}^2$ contain the horizontal and the vertical details of I at scale j. Substituting the expressions in the Wavelet Transform and simplifying gives

$$\begin{pmatrix} D_{2^j}^1 \\ \\ D_{2^j}^1 \end{pmatrix} = 2^j \begin{pmatrix} \frac{\partial}{\partial x}\left(I * \theta_{2^j}\right) \\ \\ \frac{\partial}{\partial x}\left(I * \theta_{2^j}\right) \end{pmatrix} = 2^j \nabla\left(I * \theta_{2^j}\right) \tag{5.4}$$

Thus, the wavelet transform of a gray level image has the gradient components of the image, smoothed by the dilated smoothing function θ_{2^j}.

5.3 Watershed Transform and Images

The following are the results of applying the Watershed Transform on the images of zebra, butterfly,and the fern. Watershed Transform was directly applied on the images without any preprocessing and the following results were obtained.

5.3.1 Experimentation

The following is the algorithm used for the purpose of experimenting with the Zebra Image.

1. Convert the color Zebra image to gray scaled Zebra image

2. Apply Gaussian smoothing using a range of values for standard deviation (σ)

    (a) for each σ find the 4-connected watersheds and update its image sequence

    (b) for each σ find the 8-connected watersheds and update its image sequence

    (c) for each σ find the Canny edges and update its image sequence

3. Apply Laplacian smoothing using a range of values for standard deviation

    (a) for each σ find the 4-connected watersheds and update its image sequence

    (b) for each σ find the 8-connected watersheds and update its image sequence

4. find intersection of the images in each image sequence

5.3.2 The Matlab Psuedo-Code

The following is the Matlab code used for generating image sequences scaled by the standard deviation. Intersection of each sequence of image was performed within itself to identify minimal feature detection. These features were identified at each scale. This aids in understanding the edges in the scaled Zebra image.

A similar code was used to find the intersections of all the image sequences written out to their cell matrices.

5.3.3 Watershed Transform on Zebra Without Any Preprocessing

The results of the Watershed Transform on the gray scaled images of the Zebra, the butterfly, and the ferns can be seen in Figure 5.1. It is obvious to the naked eye that the butterfly and fern are oversegmented.

Table 5.1: Matlab Psuedocode for Experimenting with the Zebra Image

```
1. Read an Image from a File
I = imread('c:\zebraedges\zebra','jpg');
global I_gray;
Convert the image into a grayscale image:
I_gray = rgb2gray(I);
[m n]= size(I_gray);

2. Perform Gaussian Smoothing on the image for a range of values
The filter used is a rotationally symmetric Gaussian lowpass
filter of size h which is a 3X3 matrix and standard deviation sigma.
sig = 0.1; j=1;
I_Gauss = cell(1,5);
I_4WSG = cell(1,5);
I_8WSG = cell(1,5);
I_CANNY = cell(1,5);
I_Canny_Intersect = ones(m,n);
I_4WG = ones(m,n);
I_4WG1 = ones(m,n);

for i = 1:1:5
    if (sig <= 300)
       h1 = fspecial('gaussian', [7 7], sig);
       I_gauss = imfilter(I_gray, h1, 'replicate', 'conv');
       I_Gauss{1,i} = I_gauss;
       I_4ws = watershed(I_gauss);
       I_4WSG{1,i} = I_4ws;
       [p q]= size(I_4ws);
       I_8ws = watershed(I_gauss, 8);
       I_8WSG{1,i} = I_8ws;
       I_canny = edge(I_gauss,'canny');
       I_CANNY{1,i} = I_canny;
       for a = 1:1:m
        for b = 1:1:n
            if I_Canny_Intersect(a,b) == I_canny(a,b)
              I_Canny_Intersect(a,b) = I_canny(a,b);
            else  I_Canny_Intersect(a,b) = 0;
            end
        end
       end
```

```
        figure, imshow(I_gauss)
        figure, imshow(I_canny)
        figure, imshow(I_4ws)
        figure, imshow(I_8ws)
        figure, imshow(I_Canny_Intersect)
        I_canny_final = edge(I_Canny_Intersect, 'canny');
        figure, imshow(I_canny_final)
        sig = sig + (1.5)^j;
    else output('Sigma is more than 300')
    end
end
```

3. Take the Laplacian of the image and apply the Watershed Transform

```
j = 0.05;
I_Laplacian = cell(1,10);
I_4WSL = cell(1,10);
I_8WSL = cell(1,10);
I_ws4 = ones(m,n);
for i = 1:1:10
    if j<=1.0
       h3 = fspecial('laplacian', j);
       I_laplacian = imfilter(I_gray, h3, 'replicate', 'conv');
       I_4wsL = watershed(I_laplacian);
       I_8wsL = watershed(I_laplacian, 8);
       I_Laplacian{1,i} = I_laplacian;
       I_4WSL{1,i} = I_4wsL;
       I_8WSL{1,i} = I_8wsL;
       j = j + 0.05;
       figure,imshow(I_laplacian)
       figure,imshow(I_4wsL)
       figure,imshow(I_8wsL)
    end
end
```

4. This code helps find Intersection of Gaussian-Smoothed Images.

```
[m n]= size(I_gray);
I_IntersectG = ones(m,n);
for j = 1:1:m
    for k = 1:1:n
        if I_Gauss{1,1}(j,k) == I_Gauss{1,2}(j,k)
           if I_Gauss{1,2}(j,k) == I_Gauss{1,3}(j,k)
              if I_Gauss{1,3}(j,k) == I_Gauss{1,4}(j,k)
                 if I_Gauss{1,4}(j,k) == I_Gauss{1,5}(j,k)
                    I_IntersectG(j,k) = 0;
                 else  I_IntersectG(j,k) = 1;
                 end
              end
           end
        end
    end
end
figure,imshow(I_IntersectG)
```

5. This code helps find Intersection of 4-Watersheds in Gaussian-Smoothed Images.

```
Rescale all grayvalues to the range: 0-255
I_4WSG1 = cell(1,5);
for i = 1:1:5
    I_WSG1 = I_4WSG{1,i};
    val = 255./(max(I_WSG1(:))-min(I_WSG1(:)));
    for x=1:1:337
        for y=1:1:503
           I_WSG1(x,y)=I_WSG1(x,y).*val;
        end
    end
    I_4WSG1{1,i}=I_WSG1;
end

I_IntersectGWS4= zeros(337,503);
    for k = 1:1:503
        for j = 1:1:337
            if I_4WSG1{1,1}(j,k) == I_4WSG1{1,2}(j,k)
                if I_4WSG1{1,2}(j,k) == I_4WSG1{1,3}(j,k)
                    if I_4WSG1{1,3}(j,k) == I_4WSG1{1,4}(j,k)
                        if I_4WSG1{1,4}(j,k) == I_4WSG1{1,5}(j,k)
                             I_IntersectGWS4(j,k) = 1;
                        end
                    end
                end
            end
        end
   end
figure,imshow(I_IntersectGWS4)
```

6. This code helps find Intersection of 8-Watersheds in Gaussian-Smoothed Images.
Rescale all grayvalues to the range: 0-255

```
I_8WSG1 = cell(1,5);
for i = 1:1:5
    I_WSG1 = I_8WSG{1,i};
    val = 255./(max(I_WSG1(:))-min(I_WSG1(:)));
    for x=1:1:337
        for y=1:1:503
           I_WSG1(x,y)=I_WSG1(x,y).*val;
        end
    end
    I_8WSG1{1,i}=I_WSG1;
end

I_IntersectGWS8= zeros(337,503);
    for k = 1:1:503
        for j = 1:1:337
            if I_8WSG1{1,1}(j,k) == I_8WSG1{1,2}(j,k)
                if I_8WSG1{1,2}(j,k) == I_8WSG1{1,3}(j,k)
                    if I_8WSG1{1,3}(j,k) == I_8WSG1{1,4}(j,k)
                        if I_8WSG1{1,4}(j,k) == I_8WSG1{1,5}(j,k)
                             I_IntersectGWS8(j,k) = 1;
                        end
                    end
                end
            end
        end
   end
figure,imshow(I_IntersectGWS8)
```

```
7. This code finds the intersection of Laplacian-smoothed images
[m n]= size(I_gray);
I_IntersectL = ones(m,n);
for j = 1:1:m
    for k = 1:1:n
        if I_Laplacian{1,1}(j,k) == I_Laplacian{1,2}(j,k)
           if I_Laplacian{1,2}(j,k) == I_Laplacian{1,3}(j,k)
              if I_Laplacian{1,3}(j,k) == I_Laplacian{1,4}(j,k)
                 if I_Laplacian{1,4}(j,k) == I_Laplacian{1,5}(j,k)
                    I_IntersectL(j,k) = 0;
                 else  I_IntersectL(j,k) = 1;
                 end
              end
           end
        end
    end
end
figure,imshow(I_IntersectL)

8. This code finds Intersection of 4-Watersheds in Laplacian-Smoothed
Images.
Rescale all grayvalues to the range: 0-255
I_4WSL1 = cell(1,10);
for i = 1:1:10
    I_WSL1 = I_4WSL{1,i};
    val = 255./(max(I_WSL1(:))-min(I_WSL1(:)));
    for x=1:1:337
        for y=1:1:503
           I_WSL1(x,y)=I_WSL1(x,y).*val;
        end
    end
    I_4WSL1{1,i}=I_WSL1;
end

I_IntersectLWS4= zeros(337,503);
    for k = 1:1:503
        for j = 1:1:337
            if I_4WSL1{1,1}(j,k) == I_4WSL1{1,2}(j,k)
                if I_4WSL1{1,2}(j,k) == I_4WSL1{1,3}(j,k)
                    if I_4WSL1{1,3}(j,k) == I_4WSL1{1,4}(j,k)
                        if I_4WSL1{1,4}(j,k) == I_4WSL1{1,5}(j,k)
                            if I_4WSL1{1,5}(j,k) == I_4WSL1{1,6}(j,k)
                        if I_4WSL1{1,6}(j,k) == I_4WSL1{1,7}(j,k)
                           if I_4WSL1{1,7}(j,k) == I_4WSL1{1,8}(j,k)
                              if I_4WSL1{1,8}(j,k) == I_4WSL1{1,9}(j,k)
                                 if I_4WSL1{1,9}(j,k) == I_4WSL1{1,10}(j,k)
                                                 I_IntersectLWS4(j,k) = 1;
                                             end
                                         end
                                     end
                                 end
                             end
                         end
                     end
                 end
             end
         end
    end
figure,imshow(I_IntersectLWS4)
```

9. This code finds Intersection of 8-Watersheds in Laplacian-Smoothed Images.

Rescale all grayvalues to the range: 0-255

```
I_8WSL1 = cell(1,10);
for i = 1:1:10
    I_WSL1 = I_8WSL{1,i};
    val = 255./(max(I_WSL1(:))-min(I_WSL1(:)));
    for x=1:1:337
        for y=1:1:503
           I_WSL1(x,y)=I_WSL1(x,y).*val;
        end
    end
    I_8WSL1{1,i}=I_WSL1;
end

I_IntersectLWS8= zeros(337,503);
    for k = 1:1:503
        for j = 1:1:337
            if I_8WSL1{1,1}(j,k) == I_8WSL1{1,2}(j,k)
                if I_8WSL1{1,2}(j,k) == I_8WSL1{1,3}(j,k)
                    if I_8WSL1{1,3}(j,k) == I_8WSL1{1,4}(j,k)
                        if I_8WSL1{1,4}(j,k) == I_8WSL1{1,5}(j,k)
                            if I_8WSL1{1,5}(j,k) == I_8WSL1{1,6}(j,k)
                                if I_8WSL1{1,6}(j,k) == I_8WSL1{1,7}(j,k)
                        if I_8WSL1{1,7}(j,k) == I_8WSL1{1,8}(j,k)
                            if I_8WSL1{1,8}(j,k) == I_8WSL1{1,9}(j,k)
                                if I_8WSL1{1,9}(j,k) == I_8WSL1{1,10}(j,k)
                                                I_IntersectLWS8(j,k) = 1;
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
figure,imshow(I_IntersectLWS8)
```

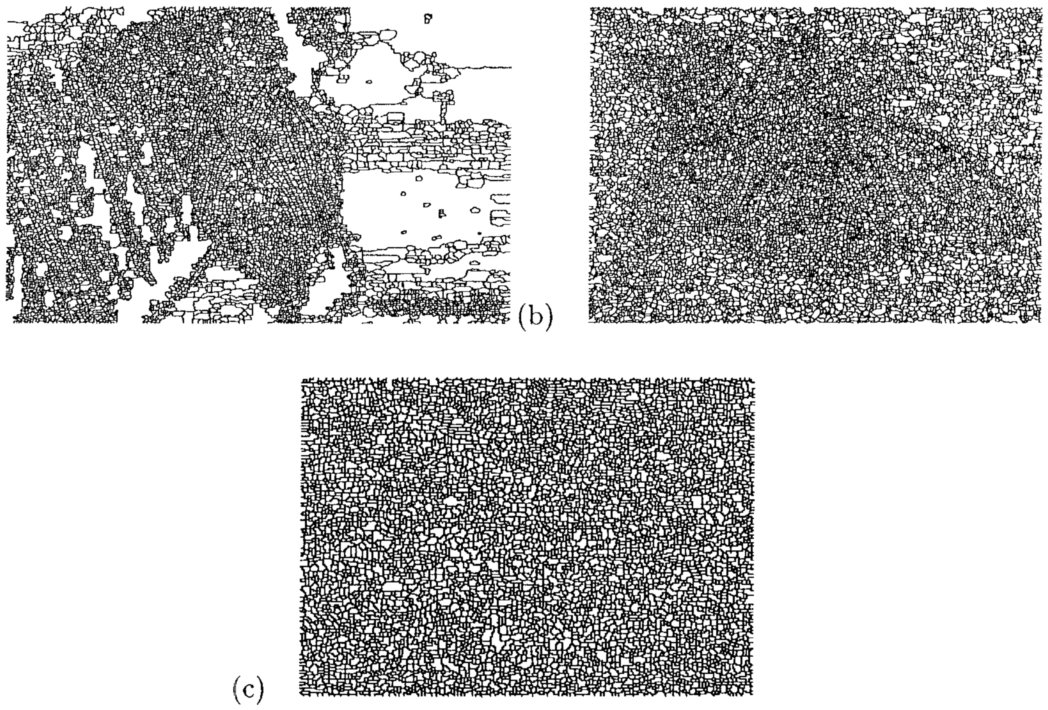
(a) (b) (c)


Figure 5.1: Watershed Transform Without any Pre-processing on (a)the Zebra, (b)the Butterfly, and (c)the Ferns

5.3.4 Watershed Transform on Zebra after Convolving with a Gaussian Filter

Following results were obtained after convolving the butterfly with a Gaussian filter of width [7 7] and varying sigma values of 0.1, 1.6, 2.35, 3.475, and 5.1625. The results can be seen in Figure 5.2.

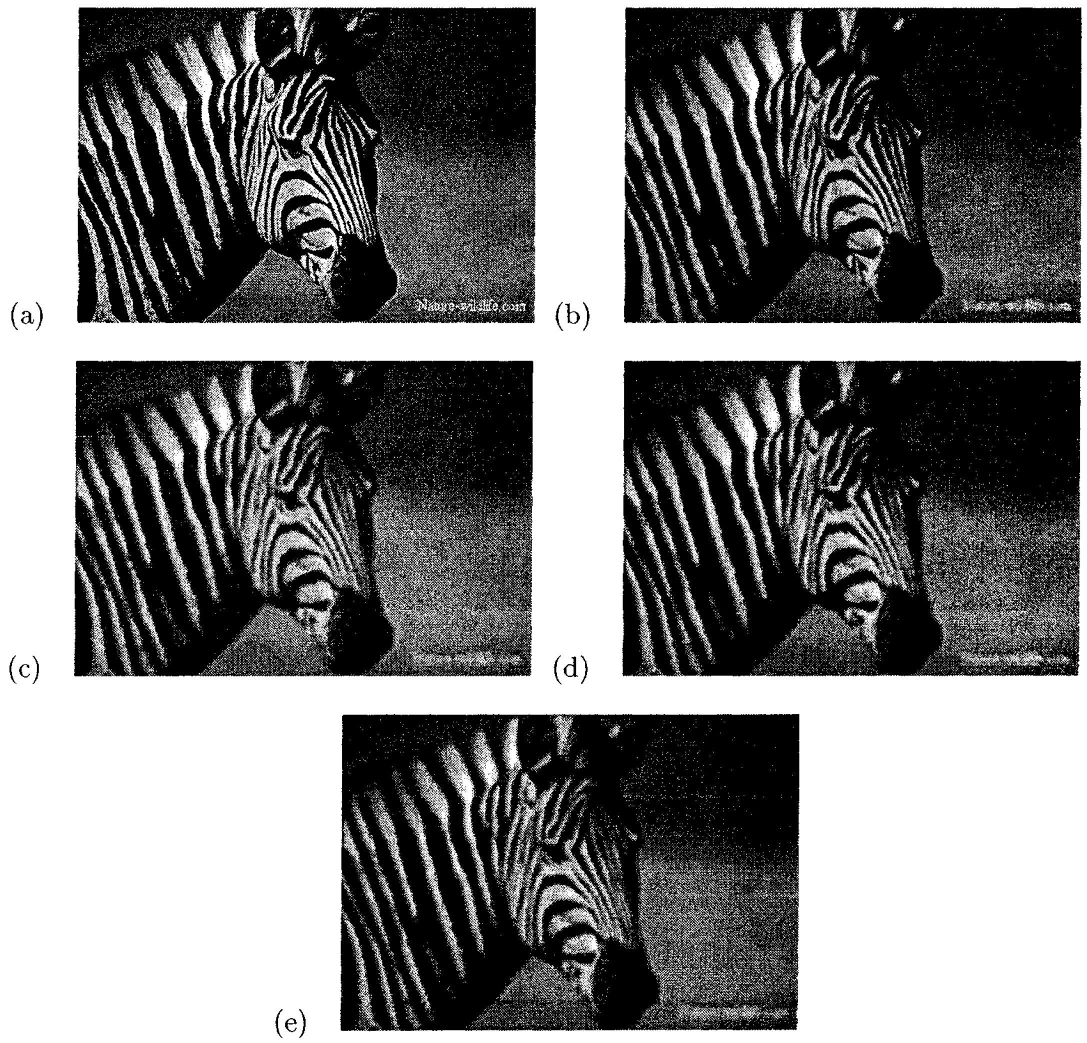


Figure 5.2: The Zebra after Gaussian Smoothing with (a)σ=0.1, (b)σ=1.6, (c)σ=2.35, (d)σ=3.475, and (e)σ=5.1625

The results of applying the Watershed Transfomation on the above gaussian smoothed Zebra images can be seen in Figure 5.3.

The loss of edges is evident with increase in the value of σ.

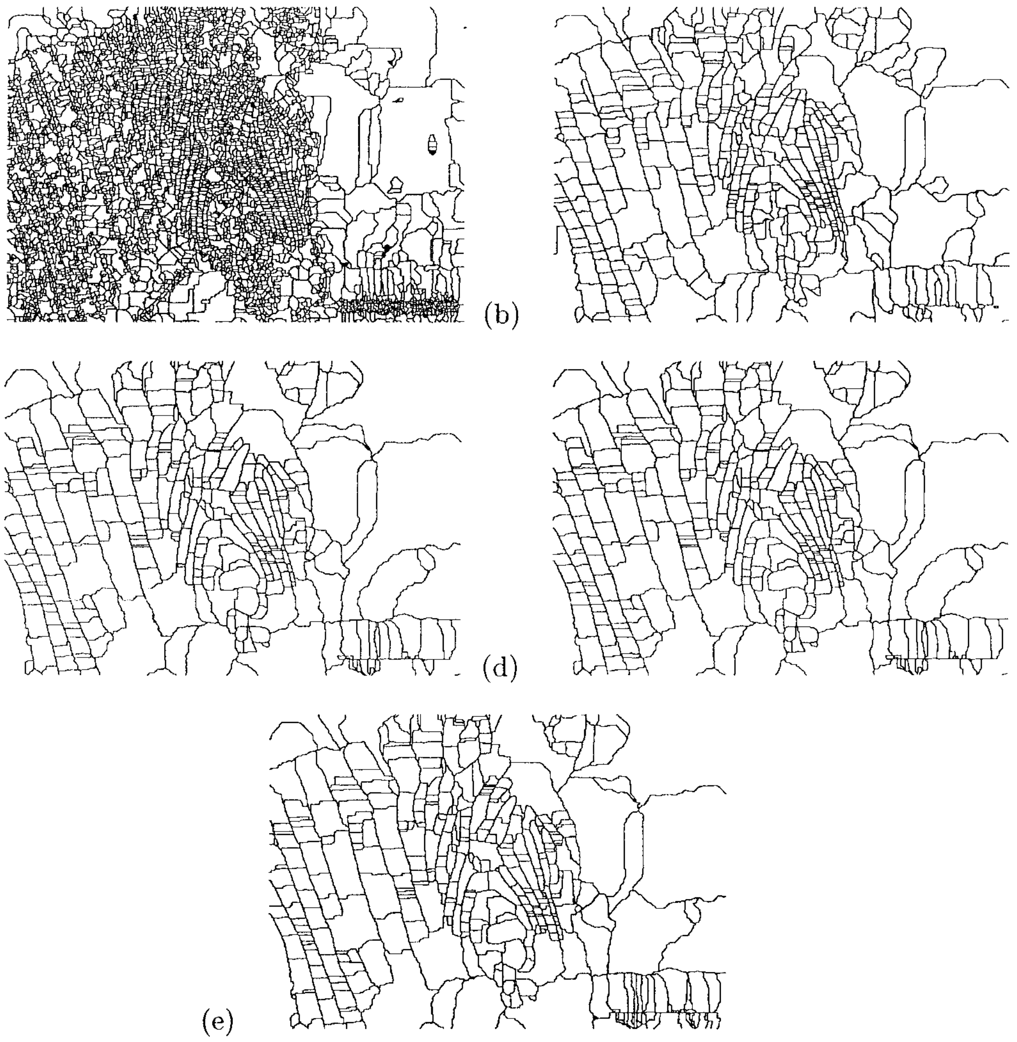
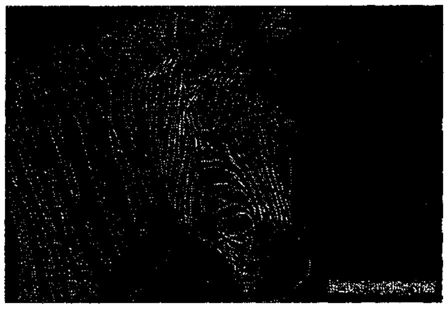
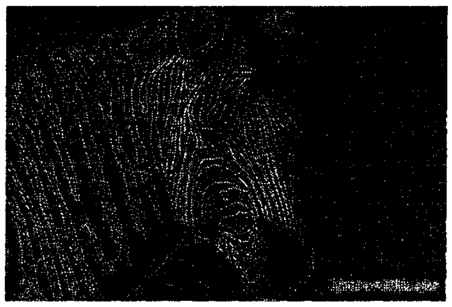
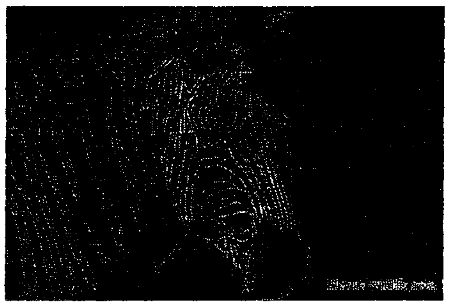


Figure 5.3: Watershed Transform Using 4-connectivity on the Zebra after Gaussian Smoothing with (a)σ=0.1, (b)σ=1.6, (c)σ=2.35, (d)σ=3.475, and (e)σ=5.1625

5.3.5 Watershed Transform on Zebra after Preprocesing with a Laplacian Filter

In this third experimentation the images were convolved with a laplacian filter of size 0.05, 0.2, 0.3, 0.4, and 0.5 value. The results obtained can be seen in Figure 5.4. The results of

(a) (b) 

(c)  (d) 

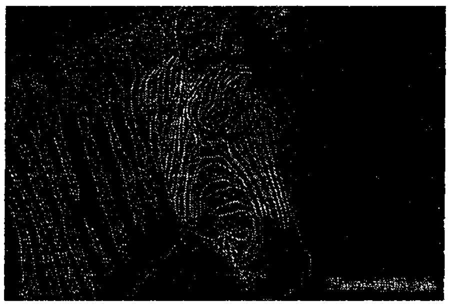
(e) 

Figure 5.4: Laplacian Operator Applied on the Zebra Image with Size: (a)0.05, (b)0.2, (c)0.3, (d) 0.4, and (e) 0.5

applying the Watershed Transform on the Laplacian treated Zebra images can be seen in

Figure 5.5. One can see improvement in the edges related to the Zebra with increase of the size of a laplacian operator. For example, see the Zebra hooves.

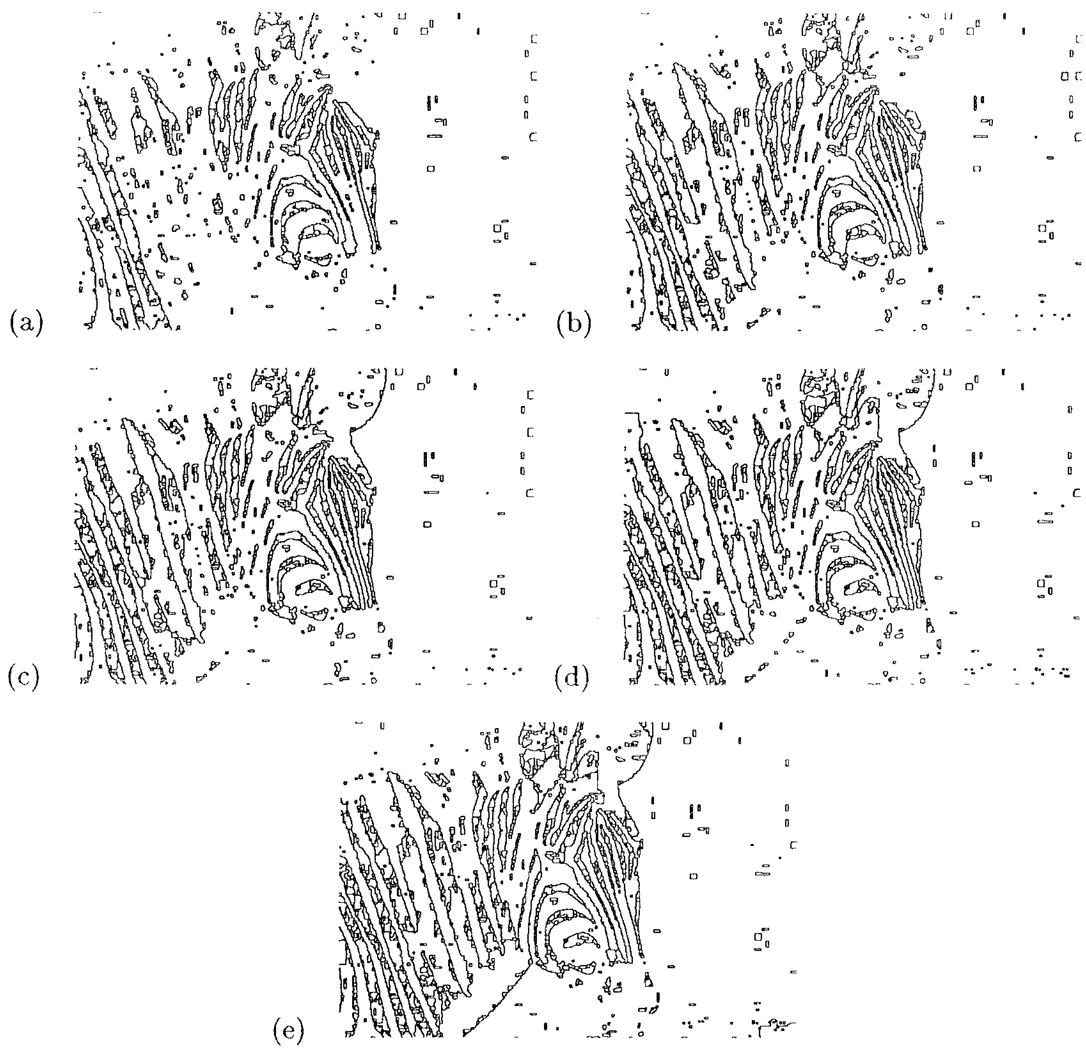


Figure 5.5: Watershed Transform Using 4-connectivity on the Zebra After Treating it with a Laplacian of Size (a)0.05, (b)0.2, (c)0.3, (d) 0.4, and (e) 0.5

5.3.6 Canny Edge Detector Applied on the Zebra Image after Gaussian Smoothing

Canny edges in the Zebra image can be seen in Figure 5.5. It is evident that by increasing the sigma values of the Gaussian filter from 0.1, 1.6, 2.35, 3.475, to 5.1625 again leads to a loss of edges. The edge-based linking method is a very powerful technique. One is tempted to prefer it over other edge detecting procedures.

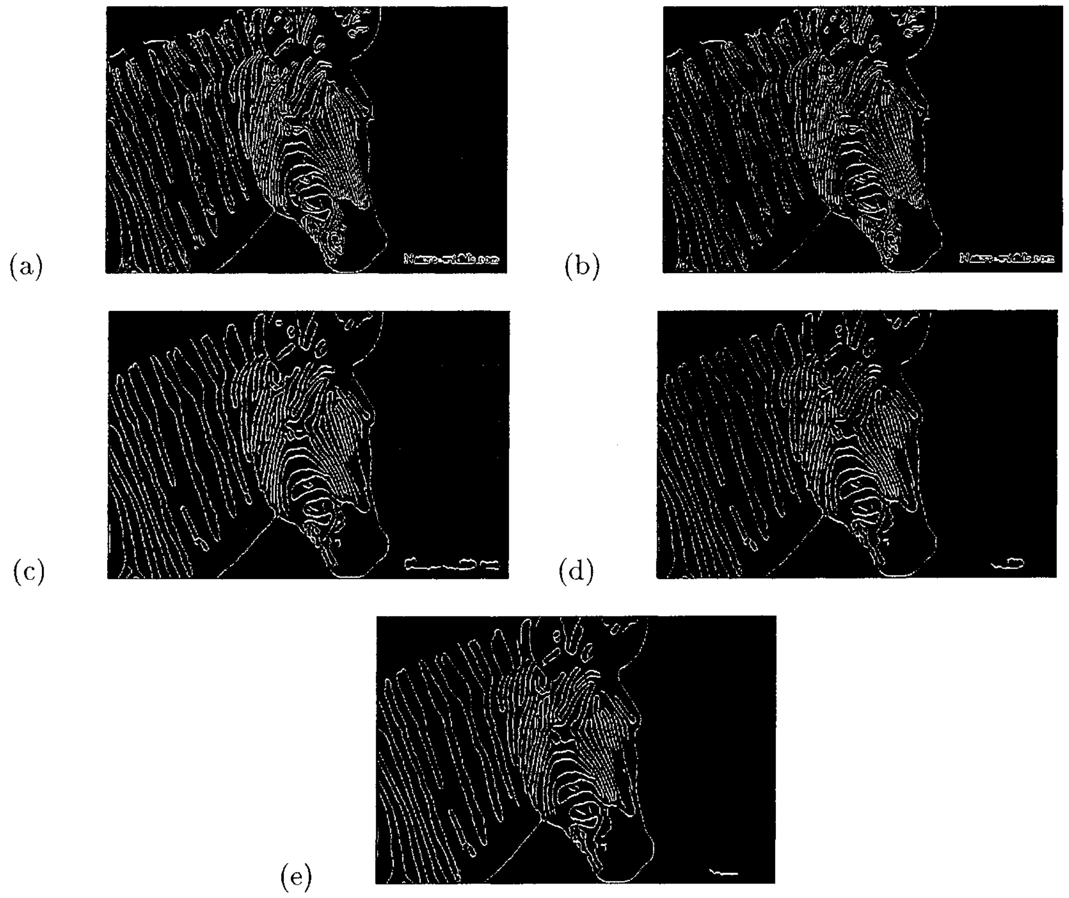


Figure 5.6: Canny Edge Detector on the Zebra After Gaussian Smoothing with (a)σ=0.1, (b)σ=1.6, (c)σ=2.35, (d)σ=3.475, and (e)σ=5.1625

5.4 The Consensus Image

The Zebra image is processed using the watershed and the various edge detection procedures in Image Processing Toolbox of Mathlab 7.0.1. The edge detection procedures applied to the images are those described in Chapter 2.

5.4.1 Intersection Operator

Operations of union and intersection of a set of images allows one to explore features common to the whole set of images. Recognition of patterns using fuzzy sets is seen in[60].

5.4.2 Intersection of the Zebra Images

Using Matlab, the gaussian smoothing and the laplacian operator were applied on the Zebra image and an array of images was generated. These were stored in a cell array structure so that they can be accessed easily. Then the Watershed Transform was applied to these images which were again stored in a separate cell array. These were successively accessed to determine the intersection of all images in a given cell array. The result can be seen in Figure 5.7.

5.4.3 Intersection of the Canny Image Set of the Zebra

The intersection of the Canny edges obtained in Figure 5.5 can be seen in Figure 5.8.

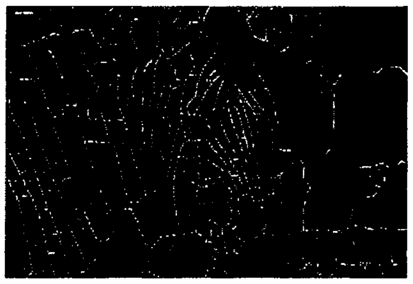
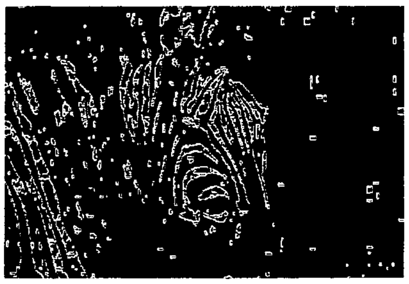
(a)  (b) 

Figure 5.7: Intersection of 4-connected Watershed Transformed Zebra Images Preprocessed with (a) the Gaussian, and (b) the Laplacian Operator

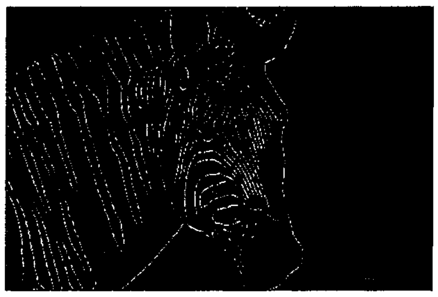


Figure 5.8: Intersection of Canny Edges in the Zebra Preprocessed with the Gaussian Filter

5.5 Summary

The Watershed Transform applied on a sequence of Gaussian-Smoothed images shows how features appear and disappear at different scales which pertain to differing values of variance used to generate them. The same can be seen when Laplacian operator was used in place of Gaussian. The above treatments were repeated by using Canny operator in place of the Watershed Transform. The results show that the Canny operator captures the edges in the Zebra more effectively, when compared to the Watershed Transform.

Watershed Transform is, theoretically, a very effective technique and is excellent in capturing just about all the edges in an image. But when it comes to relating the edges to the desired features, it fails to some extent. The various preprocessing, and postprocessing techniques were tried out in an attempt to understand where and what exact treatment can help improve the watersheds. An intersection of the watershed images from the set of gaussian- and laplacian-treated Zebra image goes on to prove that two procedures do not aid in comparing the edges of the Zebra as a whole.

Chapter 6

Conclusion

This thesis takes a look at the various techniques that affect the results of the Watershed Transform on the Zebra image. The key operators studied and experimented with are the Morphological, the Gaussian, and the Laplacian operators.

Chapter 4 provides an insight as to the effect of the morphological operators on the image and hence aids in improvizing the Watershed Algorithm itself. This also helps in better ordering the techniques or the operators that one would need in order to study or extract the features or objects of interest.

Chapter 5 takes a look at the gaussian and the laplacian smoothing of the Zebra image and how the watersheds are affected. One can see that in these image sequences the details appear and disappear with the varying degrees of smoothness. In particular, small features are lost at lower levels of smoothing while large features are lost at higher levels. This implies that the degree of smoothing can be linked to the size of a feature or an object in an image. So, by looking at the whole spectrum of images in an image sequence one can identify the image/s which are more informative, and hence the level of the scaled-space used to replace the original image.

The Watershed Transform is applied to the preprocessed Zebra image. A very careful extraction of the Zebra edges is shown in Chapter 4 which proves that the process is less automatic and requires user intervention for a particular feature extraction. Chapter 5 attempts to automate the process but in doing so one is caught at the level or levels into which the feature is contained. This again warrants manual intervention for a feature study.

All the above suggest that even though the Watershed Transform is efficient in detecting closed boundaries, user-intervention is inevitable to make these contours more meaningful. One major reason for this is the problem of oversegmentation in the watershed results which we try to minimize by experimenting with different operators.

The use of wavelets and a scale-space to represent an image [61] and then study the watersheds therein is gaining popularity. Because of the time constraint this needs to be pursued in a future research.

References

[1] J.B.T.M. Roerdink, and A. Meijster, *The Watershed Transform: Definitions, Algorithms, and Parallelization Strategies*, in Fundamentae Informaticae 41(2000) 187-228, IOS Press.

[2] S. Buecher, and F. Meyer, *The Morphological Approach to Segmentation: The Watershed Transformation*, In Mathematical Morphology in Image Processing, E.R. Dougherty, ED., Marcel Dekker, New York, 1993, ch. 12, pp. 433-481.

[3] Rafael C. Gonzalez, and Richards E. Woods, *Digital Image Processing*, Second Edition, Prentice Hall, Upper Saddle River, New Jersey 07458.

[4] Laurent Najman, and Michel Schmitt, *Watershed of a Continuous Function*, in Signal Processing 38 (1994) 99-112.

[5] Laurent Najman, and Michel Schmitt, *Geodesic Saliency of Watershed Contours and Hierarchical Segmentation*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 12, December 1996, pp. 1163-1173.

[6] Jaesang Park, and James M. Keller, *Snakes on the Watershed*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 10, October 2001.

[7] Fernand Meyer, *Topographic Distance and Watershed Lines*, in Signal Processing, Vol. 38, 1994, pp. 113-125.

[8] Francis Friedlander, Fernand Meyer, *A Sequential Algorithm for Detecting Watersheds on a Gray Level Image*, in ACTA Sterol 1987; 6/III: 663-668, PROC ICS VII CAEN 1987, original scientific paper.

[9] Stanislav L. Stoev, and Wolfgang StraBer, *Extracting Regions of Interest Applying a Local Watershed Transformation*, in PROC IEEE Visualization 2000, 8-13 Oct 2000, Salt Lake City.

[10] P. Smet, and R.L. Pires, *Implementation and Analysis of an Optimized Rainfalling Watershed Algorithm*, Internal Report TELIN/IPI (TWO7V), University of Ghent, available at http://telin.rug.ac.be/ipi/watershed/.

[11] D. Vleeschauwer, P. Smet, F. Cheikh, R. Hamila, M. Gabbouj, *Optimal Performance of the Watershed Segmentation of an Image Enhanced by Teager Energy Driven Diffusion*, internal report TELIN/IPI, University of Ghent, Belgium.

[12] P. Scheunders, *Multiscale Fundamental Forms: a Multimode Image Wavelet Representation*, Vision lab, Department of physics, University of Antwerp,Groenenborgerlaan 171, 2020 Antwerp, Belgium, scheun@ruca.ua.ac.be.

[13] Richard M. Beam, and Robert F. Warming, *Multiresolution Analysis and Super Compact Multiwavelets.*

[14] S. Mallat, and S. Zhong, *Characterization of Signals from Multiscale Edges*, IEEE,Trans. Pattern Anal. Machine Intell., 14:710-732, 1992.

[15] P. Scheunders, and J. Sijbers, *Multiscale Watershed Segmentation of Multivalued Images*, 2002 IEEE, 1051-4651/02 17 ~(c) .

[16] Mark A. Pinksy, *Introduction to Fourier Analysis and Wavelets*, The Brooks/Cole Series in Advanced Mathematics, Paul J. Sally, Jr., Editor.

[17] Robi Polikar, *The Wavelet Tutorial*, www.users.rowan.edu.

[18] Dorin Comaniciu, *Image Segmentation using Clustering with Saddle Point Detection*, www.caip.rutgers.edu/ comanici/Papers/ImageSegmentationClustering.pdf.

[19] Bill Green, *Canny Edge Detection Tutorial*, www.pages.drexel.edu\ ~weg22/cana_tut.html.

[20] Tonette R. M. King, *Efficient and Effective Methods for Object Segmentation in Video Images.*

[21] J. Canny, *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp 679-698, Nov 1986.

[22] Tai Sing Lee, *Canny Edge*, CMU 15-385 Computer Vision, Spring 2002.

[23] Michael McAllister, Jack Snoeyink, *Extracting Consistent Watersheds from Digital River and Elevation Data*, Department of Computer Science, University of Columbia.

[24] Michael McAllister, *A Watershed Algorithm for Triangulated Terrains*, University of British Columbia.

[25] D. Mann and E. Hildreth, *Theory of Edge Detection*, Proc. Roy. Soc. Lond. B, 207,187-217 (1980).

[26] M. Sezgin, and B. Sankur, *Survey Over Image Thresholding Techniques and Quantitative Performance Evaluation*, J. Electronic Imaging, 13, (2004), 146-165.

[27] C. Kunos, L. Latson, B. Overmoyer, P. Silverman, R. Shenk, T. Kinsella, J. Lyons, *Breast Conservation Surgery Achieving Greater or Equal 2mm Tumor-free Margins Results in Decreased Local-Regional Recurrence Rates*, J. Breast, $\underline{12}$, (2006) 28-36.

[28] S.M. Smith, and J.M. Brady, *SUSAN - A New Approach to Low Level Image Processing*, Int, Journal of Computer Vision, $\underline{23}$,(1997), 45-78.

[29] J. Shen, and S. Castan, *An Optional Linear Operator for Step Edge Detection*, Computer Vision, Graphics and Image Processing: Graphical Models and Understanding, $\underline{54}$,(1992), 112-133.

[30] Charles R. Giardina, and Edward R. Dougherty, *Morphological Methods in Image and Signal Processing*, Prentice Hall.

[31] M. Razaz, D.M.P. Hagyard, *Morphological Segmentation of Multidimensional Images*, School of Information Systems, University of East Anglia, Norwich, England.

[32] A.K. Jain, *Image Processing.*

[33] C. Riddel, P. Brigger, R.E. Carson, and S.L. Bacharach, *The Watershed Algorithm: A Method to Segment Noisy PET Transmission Images*, In IEEE Transactions on Nuclear Science, vol.46, no. 3, June 1999, pp. 713-719.

[34] E.N. Mortensen, and W.A. Barret, *Toboggan-based Intelligent Scissors with a Four-parameter Edge Model* in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1999, pp.452-458.

[35] M. Couprie, L. Najman & G. Bertrand, *Quasi-linear Algorithms for the Topological Watershed*, J. Math. Imaging and Vision, **22** 231-249 (2005).

[36] J. Cousty, G. Bertrand, M. Couprie, L. Najman, *Fusion Graphs: Merging Properties and Watersheds*, preprint (2006).

[37] D. Gabor, *Theory of Communication*, J. IEEE, **93** 429-457a(1946).

[38] Y. Lin, Y. Tsai, & Z. Shih, *Comparison between Immersion-based and Toboggan-based Watershed Image Segmentation*, IEEE Trans. Image Processing, **15** 632-640 (2006).

[39] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, London (1998).

[40] G. Matheron, *Random Sets and Integral Geometry*, Wiley, New York (1950).

[41] F. Meyer, & S. Beucher, *Morphological Segmentation*, J. Vis. Comm and Im. Rep. **1** 21-46 (1990).

[42] O. Monga, *An Optimal Region Growing Algorithm for Image Segmentation*, Int. J. Patt. Recog. Artificial Intell. **3** (1987).

[43] L. Najman, M. Couprie, G. Bertrand, *Watersheds, Mosaics and the Emergence Paradigm*, Discrete Appl. Math. **147** 301-324 (2005).

[44] H.T. Nguyen, M. Worring, & van den Boomgaard, R., *Watersnakes: Energy-driven Watershed Segmentation*, Preprint Intelligent Sensory Inf. Systs., Univ. Amsterdam, Netherlands, (2001).

[45] A. Rosenfeld, *Connectivity in Digital Pictures*, J. Assn. for Computer Machinery, **17** 146-160 (1970).

[46] A. Rosenfeld, *On Connectivity Properties for Grayscale Pictures*, Pattern Recognition, **16**, 47-50 (1983).

[47] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press London, 1982.

[48] J. Serra, *Introduction to Mathematical Morphology*, Comp. Vis. Graph. Image Process. **35** 283-305 (1986).

[49] L. Vincent, and Beucher, *The Morphological Approach to Segmentation : An Introduction*, Technical Report CMM, School of Mines, Paris (1989).

[50] L. Vincent, and P. Soille, *Watersheds in Digital Spaces: An Efficient Algorithm based on Immersion Simulations*, IEEE Trans. Patt. Anal. and Mach. Intell. **13** 583-598 (1991).

[51] J. Serra, & L. Vincent, *An Overview of Morphological Filtering*, Circuits, systems and signal proc., **11** 47-108 (1992).

[52] Luc Vincent, *Morphological Grayscale Reconstruction: Definition, Efficient Algorithm and Application in Image Analysis*, Proc. IEEE Conf. on Comp. Vision and Pattern Recog. Champaign IL, 633-635 (1992).

[53] Luc Vincent, *Morphological Grayscale Reconstruction in Image Analysis: applications and efficient algorithms*, IEEE Trans. Im. Proc., **2** 176-201 (1993).

[54] D. Wang, *Unsupervised Video Segmentation based on Watershed and Temporal Tracking*, IEEE Trans. Circuits and systs. for video tech. **8**, 539-546 (1998).

[55] S. Marshall, and E.R. Dougherty, *New Challenges in Non-Linear Signal and Image Processing*, http://www.eurasip.org/content/Eusipco/2004/defevent/papers/cr1135.pdf.

[56] K. Rank, M. Lendl, R. Unbehauen,*Estimation of Image Noise Variance*, IEE Proc.-Vis. Image Signal Process., Vol. 146, No. 2, April 1999, pp.80-84.

[57] C. Liu, W.T. Freeman, R. Szeliski, S.B. Kang, *Noise Estimation from a Single Image*, to appear at IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[58] V. Grau, A.U.J. Mewes, M. Alcaniz, *Improved Watershed Transform for Medical Image Segmentation Using Prior Information*, IEEE Transactions on Medical Imaging, Vol. 23, No. 4, April 2004, pp.447-458.

[59] W. Bieniecki, *Oversegmentation Avoidance in Watershed-Based Algorithms for Color Images*, TCSET2004, February 24-28, 2004, Lviv-Slavsko, Ukraine, pp 169-172.

[60] B. Shukat, *Labels Evaluation for the Fuzzy Patterns Recognition*, 0-7803-3225-3-6/96, 1996 IEEE.

[61] Peter Majer, *A Statistical Approach to Feature Detection and Scale Selection in Images*, Dissertation, Gottingen, Mai 2000.

[62] A. Moga, *Parallel Watershed Algorithms for Image Segmentation*, PhD Thesis, Tampere University of Technology, Tampere, Finland, 1997.