IMAGE PROCESSING BY REGION EXTRACTION USING A CLUSTERING APPROACH BASED ON COLOR

Kam Shek Simon Leung

Department of Computing Science
University of Stirling
January 1991

A thesis submitted in partial fulfilment of requirements
for the Degree of Doctor of Philosophy
of the University of Stirling

Abstract

This thesis describes an image segmentation technique based on watersheds, a clustering technique which does not use spatial information, but relies on multispectral images. These are captured using a monochrome camera and narrow-band filters; we call this color segmentation, although it does not use color in a physiological sense. A major part of the work is testing the method developed using different color images.

Starting with a general discussion of image processing, the different techniques used in image segmentation are reviewed, and the application of mathematical morphology to image processing is discussed. The use of watersheds as a clustering technique in two-dimensional color space is discussed, and system performance illustrated. The method can be improved for industrial applications by using normalized color to eliminate the problem of shadows. These methods are extended to segment the image into regions recursively. Different types of color images including both man made color images, and natural color images have been used to illustrate performance. There is a brief discussion and a simple illustration showing how segmentation can be used in image compression, and of the application of pyramidal data structures in clustering for coarse segmentation.

The thesis concludes with an investigation of the methods which can be used to improve these segmentation results. This includes edge extraction, texture extraction, and recursive merging.

Acknowledgements

I am extremely grateful to my supervisor, Dr. L.S. Smith, for his assistance, enthusiasm, criticism and encouragement during the course of the work described in this thesis.

I would also like to acknowledge Dr. R.J. Watt and Dr. A.I. Watson for a lot of stimulating and useful discussion.

My thanks to P. Hancock for his constant help in solving the programming and equipment problem.

My thanks to G. Cochrane and S. Nelson for their technique supports.

Thanks are also due to my family for all their encouragement.

Contents

List of Figures

Chapter 1

INTRODUCTION TO IMAGE PROGRESSING

1.1 Introduction

Image processing can be divided into two areas, namely analog and digital. Analogue image processing is concerned with image signals which vary continuously in intensity, and space, but digital image processing is concerned with images that are quantised with respect to intensity and space, so that they can be processed using digital computer. Video signal processing can also be considered as one kind of image processing. The difference is that video signals vary with time as well. An image is a two-dimensional light intensity function f(x,y), where (x,y) denotes spatial coordinates and the value of f at any point (x,y) is proportional to the brightness of the image at that point. A digital image is an image where both spatial coordinates and brightness have a discrete values. The points forming a digital image are termed pixels, and the discrete brightnesses are termed gray-levels.

Image processing is concerned with the manipulation and analysis of images by com-

puter. It can be classified into three major subareas [Rosenfeld & Kak 82]:

1. Image digitization, compression and coding.

2. Image enhancement, restoration and reconstruction.

3. Image matching, description and recognition.

1.2 Image Digitization, Compression and Coding

Digitization is the process of converting an image generated by a camera from its original analog form into digital form. This process entails analog-to-digital conversion (ADC). Since digital computers process finite-length numbers, it is necessary to reduce the continuous measurement value to discrete units and represent them appropriately. There are two components to be considered in this process: spatial quantisation and gray-level quantisation. Both are illustrated in figure(1.1). Each of these methods are discussed in more detail below.

1. Spatial quantisation: Spatial quantisation corresponds to sampling the brightness of the image at a number of points: for example an $M \times N$ array of points, where M and N are the number of pixels in the horizontal and vertical direction. Usually it is done in a rectangular grid and the value is the gray-level at the sampling point (figure(1.2)). The image is then represented as an array of pixels, representing samples of the image brightness at a grid of points. The grid of brightness samples must be finer than the smallest features of interest in the image. Otherwise it may miss a feature completely. It is obvious that the approximation improves as M and N increases. However, the original image can be reconstructed exactly from the digitised image as long as the sampling frequency is at least twice the highest

frequency present in the image. This 'sample at twice the maximum frequency' rule is known as Shannon's sampling theory and the rate of sampling is called Nyquist frequency. A good presentation of sampling theorem may be found in [Rosenfeld & Kak 82].

2. Gray-level quantisation: Since the gray-level at the sampling points may take any value in a continuous range, for digital processing the gray-level needs to be quantized. In gray-level quantization, the range of gray-level is divided into K intervals, and the gray-level at any point is required to take on only one of these integer values. These values are then coded and represented as binary numbers. The number of bits required will depend on the number of gray-levels. For example, 256 gray-levels will require 8 bits. In an image comprising K gray-levels, level zero normally represents black, and level $K-1$ normally represents white.

A digital image may, thus, be considered as a matrix whose row and column indices identify a pixel in the image and the corresponding matrix element value identifies the gray-level at that point. In order for the reproduced picture to be a 'good' reproduction of the original, M, N and K have to be large. Ordinarily, the finer the sampling and quantization, the better the reproduced image. However, nothing is gained by increasing M, N, and K beyond the spatial and gray scale resolution capabilities of the image acquisition system.

The transmission and storage of pictorial information in digital form is of practical importance in diverse application areas, including digital television, picture-phone, remote sensing etc., mainly because digital communication systems are more flexible and controllable. The central problem in digital image communication is channel capacity or data volume reduction while maintaining an acceptable image quality. Image compression and coding have been used to reduce the total number of bits required to code an image by

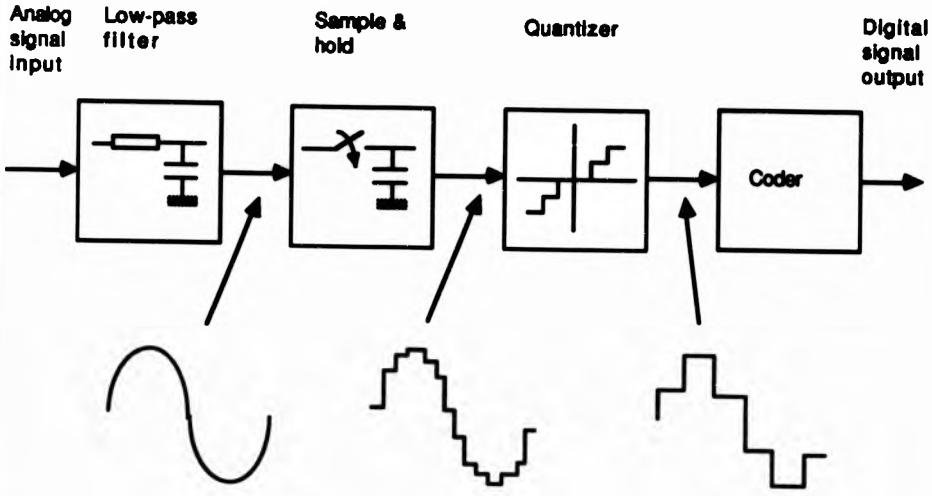


Figure 1.1: Illustration of analog-to-digital conversion, the low-pass filter is used to supress the high frequency noise (i.e. noise at a frequency more than 2 x sampling frequency) from the input analog signal.

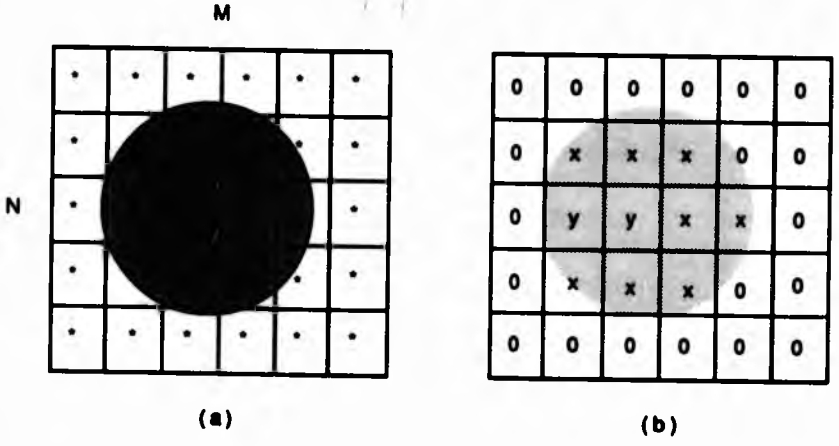


Figure 1.2: (a) Spatial quantisation of an image, where * represents a sampling point, and (b) x and y represent two different gray-levels sensed at the sampling points.

removing the redundant or irrelevant information from the image. There are two types of technique for this:

1. Reversible image compression and coding.

2. Nonreversible image compression and coding.

Reversible coding techniques attempt to compress the image data in a reversible way, so reconstruction of the original image with no degradation can been achieved. This can be done by removing redundant information in the image subject to Shannon's information theory [Shannon & Weaver 49]. The basic idea is that since not all the gray-levels in an image occur equally often, by assigning shorter binary code words to the more frequently occurring gray-level one can achieve compression over the case when all the levels are represented by equal length binary code words. The examples of this method are Huffman coding and Run length encoding [Hall 79].

For many application such as video-conferencing and deaf communication [Pearson & Robinson 85], the quality requirement for the image is not so important, so nonreversible coding can be employed to allow more compression. An information theory guideline for the nonreversible coding problem is Rate-distortion theory, which allows a trade-off between the achievable coding rate and signal distortion. The theory is extensively discussed in the book by [Berger 71], and [Gallager 68].

1.3 Image Enhancement, Restoration and Reconstruction

In image enhancement, the objective is to process a given image so that the result is more suitable than the original image for a specific application. For example, increasing the contrast of a image is a reasonable enhancement operation for human viewing. In image

restoration, an ideal image has been degraded and the objective is to make the processed image resemble the original as much as possible. However, image enhancement is closely related to image restoration. When an image is degraded, restoration of the original image often results in enhancement. The main difference is that an original, undegraded image cannot be further restored but can be enhanced. For example, high-pass filtering, where the low frequency components are reduced and the higher frequency components are preserved, produces sharpened contrast and better edges, since the edges are associated with high frequencies [Gonzalez & Wintz 87].

Enhancement methods include histogram equalisation, edge sharpening and pseudo-color enhancement. Restoration methods include geometric transformations, filtering and deblurring [Gonzalez & Wintz 87]. It is not necessarily to have a clear boundary separating the two areas, since some techniques can be applied in both.

Image reconstruction from sets of projections is sometimes necessary, since most physical objects are spatially three dimensional, and the object characteristics may only be measured after the three-dimensional reconstruction has been formed [Castleman 79]. The basic techniques for reconstructing a three-dimensional object from its two-dimensional projection are of wide application. The significance of this technique is shown by its wide application to medical areas such as computed tomography, where X rays are used to generate projection data for a number of cross sections of the human body. From the projection data a cross-sectional image depicting the morphological details of the body in that cross section can be constructed. Due to the radiation hazards associated with x rays, computed tomographic imaging with ultrasound attracts considerable attention. However, the later method can only be suitable for soft tissue structures because of the highly distortion caused by beam refraction in the presence of bone [Kak 84].

1.4 Image Description, Matching and Recognition

Ideally, image descriptions should be independent of object size, orientation and location and should contain enough discriminatory information to uniquely identify one object from another. Since a region of interest can be described by the shape of its boundary or by its internal characteristics, we can subdivide description into two principal categories:

1. Boundary descriptors; represent the region based on its external characteristics (i.e., its boundaries),

2. Regional descriptors; represent the region based on its internal characteristics (i.e., the pixels comprising the region).

Generally, an external description is chosen when the primary focus is on the shape characteristic (also called Morphological features), while an internal description is selected when one is interested in internal properties, such as color and texture.

Image descriptions generally specify properties of parts of the image and relationships among these parts. Thus such descriptions are often represented by relational structures such as graphs in which the nodes correspond to the parts; each node is labeled with its associated property value such as shape, size, color or texture; and the arcs correspond to relations between parts, labeled with the associated relation and with a value as well if qualitative. In the special case where the description does not refer to parts of the image, it consists simply of a list of property values defined for the image as a whole.

In image matching, the tasks involve comparing the given image with a standard reference and verifying the existence of discrepancies. Semantic knowledge of the relationship between the each part may need to be identified, unless the image is extremely simple and heavily constrained so that object matching processes can be applied directly to the

image. A more flexible approach is to measure a set of properties of the image and to compare the measured values with the corresponding expected values. An example of this is the use of width measurements to detect flaws in printed circuit boards. The connectors on the boards should have standard width. When a narrow one is found, it may be an indication of possible defects. There are many situations in which we want to match or register images with each other, or match some given pattern with a image. The following are some common examples:

1. Given two or more images of the same scene taken by different sensors, if they can be brought into registration with one another, we can determine the characteristic of each pixel with respect to all of the sensors.

2. Given two registered images of a scene taken at different times, the points at which they differ can be determined, and thus the changes that have taken place can be analysed. For example, in remote sensing, changes in land utilization can be detected.

3. Given two images of a scene taken from different positions, if we can identify the corresponding scene points in the two images, we can determine their distances from the camera by triangulation and thus obtain three-dimensional information about the scene. This process is known as stereomapping [Baker 80]. The main difficulty with this approach is that it is not always easy to find pairs of corresponding points.

4. Given a pictorial description of a region of a scene, we may want to determine which region in another image is similar. The simplest approach is called template matching. The template is, in effect, a subimage that looks just like the image of the object. A similarity measure is computed which reflects how well the image data

matches the template for each possible location. The point of maximal match can be selected as the location of the feature. This method can be used in detection of a known object and is used in applications, such as automated inspection of integrated circuits and printed circuit boards [Hara et al 83].

The eventual goal of recognition, which might be dependent on matching the data with stored models, is a labeling, where the function of recognition algorithms is to identify each segmented object in a scene and to assign a label (e.g., sea, sky) to that object [Ballard & Brown 82][Gonzales 74]. An example of a simple application is the automatic analysis of cells where the problem might be to locate cells in an image and to measure simple parameters such as the number and size of the cells.

Recognition approaches can be divided into two principal categories of decision-theoretic (statistical) and syntactic (structural) [Fu et al 80].

1. Decision-theoretic methods are based on quantitative descriptions. Usually two stages are required (figure(1.3)): the first is extraction in which a set of characteristic measurement, called features, are extracted from the image; the second is classification in which the recognition of each pattern is made by partitioning the feature space. Thus this method is good for measurements that can be well represented in feature space (for example, statistical texture); the structural information about the image is not considered important.

2. The syntactic method is to divide a picture into simpler 'subpictures', and then each subpicture is treated as a new picture and divided again into even simpler subpictures, and so on. The final subpictures are called 'picture primitives'. These primitives will be linked together by a set of grammatical rules to produce 'sentences'.

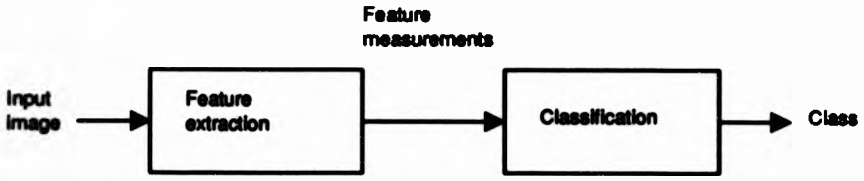
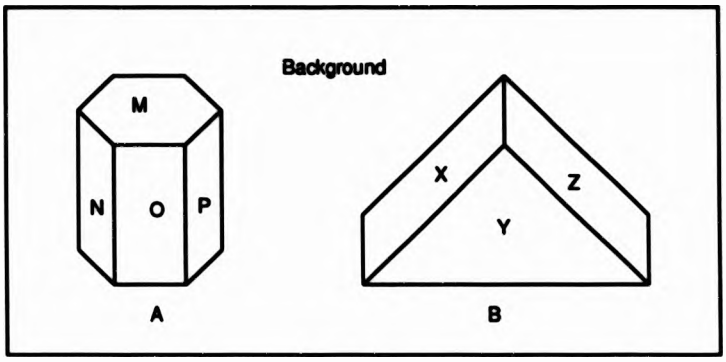


Figure 1.3: A two-stage pattern recognition system.

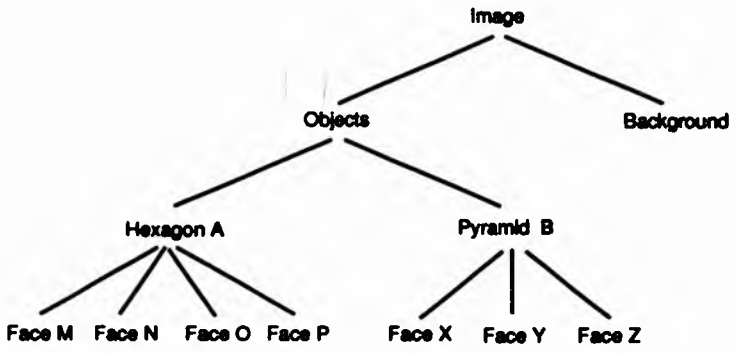
The decomposition of the image is then described in terms of a 'picture descriptive language'. Object recognition is then accomplished by performing a syntax analysis of the sentence describing the given picture, rather than directly from the input image. Thus a syntactic approach to pattern recognition is a process in which structural descriptions and their relationships are compared to models to find the most similar model. Therefore this method is good for images whose structural properties are pre-dominant in their descriptions (for example, in fingerprint and Chinese character identification, etc.). An extensive discussion in this approach can be found in [Fu 82].

This type of recursive sub-division can be demonstrated by using the building block image shown in figure(1.4a), which shows a simple building block image composed of a hexagon A and a pyramid B. Face M,N,O and P are parts of hexagon A. Face X,Y and Z are parts of pyramid B. Hexagon A, and pyramid B together with the background constitute the whole image. The structural representation of the image using tree description is illustrated in figure(1.4b).

The applications of image recognition vary from 'simple' character recognition, where an image of page is transformed into machine-readable text, to complex scene matching for medical diagnosis where the size of a lung or its texture might be obtained in a chest



(a)



(b)

Figure 1.4: (a) Building block image, and (b) structural representation of the image using tree description.

X-ray image, industrial inspection where the position of faults in manufactured articles would be pin-pointed, or earth resource studies from satellite images.

The selection of approach depends primarily on the nature of the data samples involved in a problem. Since there are many problems which have both properties, a combination of these two approaches may result in a more efficient and practical scheme for solving the problem.

1.5 Image Segmentation

Two types of segmentation have been defined [Levine 80]. The first is complete segmentation, in which the object model must in general be taken into account in order to achieve a final result consisting of the segmented regions which correspond exactly to the objects in the picture. The second is partial segmentation, in which only the local predicates of the image such as intensity, texture and color are used. This is a general purpose approach in the sense that it segments the input image into a set of coherent regions based on the local predicate used without using any specific knowledge about the image domain, application or problem that is to be solved; and each picture is treated nearly identically. Usually, the segmented images are somewhat noisy, and do not correspond perfectly to semantically significant objects.

However, they are the atomic regions that, if properly joined, will give us a structural description of the original images. So complete segmentation can be achieved using the segments, which are created in the partial segmentation, as a basis for further analysis at a higher level [Levine 78] [Nazif & Levine 84]. In the higher level stage a priori knowledge about the image is utilized. It usually involves semantics about the class of image being processed. Moreover, the analysis deals with a representation of the features extracted

from an image (i.e., segments) rather than with the image itself (i.e., pixels). The computational processing could be simplified, since the number of segments is much smaller than the number of pixels in the original image. In other words, the goal of data compression has been achieved after partial segmentation.

In the past, most efforts in image segmentation have been concerned with black-and-white images, since they are easier to handle. But all actual scenes and images contain color, and color information plays important roles in human perception and recognition of scenes and images. Of course there are some cases where color is not intrinsically related to the object's identity in the way that other cues, such as shape, are. For example, it may be nonsense to model a car as being red and green, but this color information should be very useful in locating it among the other cars unless the others are the same color. In other words color not only increases the perceptibility of important details but also increases the speed of object recognition. This explains why the pseudocolor enhancement is used to present numerical images, such as satellite images, in a chromatic manner. This also suggests the importance and usefulness of color information in image segmentation.

One of the reasons for the limited use of color images may lie both in the low availability or accessibility of specific hardware and software for acquisition and processing of color images. And the increase in complexity and computation time caused by the use of colors is quite notable. However, the situation has changed sharply due to the greater availability of acquisition and processing hardware for color images such as CCD color TV-cameras, frame stores, very-large-scale integration circuit (VLSI) and transputer, etc. As a result of more hardware becoming available, the availability of software is also increased. In other words, color should no longer be regarded only as a means of improving the beauty of a B/W image, but rather as an aid to help us to solve complex segmentation problems.

1.6 Outline of the Thesis

In this thesis, clustering and recursive clustering techniques using watersheds algorithms are being used as a partial segmentation method for color images. A considerable amount of research into image segmentation using recursive region splitting based on color has been done by many authors. For example, [Ohlander et al 78] used nine one-dimensional histograms of color features in recursive histogramming. [Ohta et al 80] examined the segmentation in natural scenes using different color features, and suggested that three effective color features can be obtained using a linear transformation of the R,G, and B color space. [Shafer & kanade 82] presented an improved version of the recursive regions segmentation by analysis of histograms of color. [Tominaga 88] introduced a method for segmenting a color image into uniform color regions by means of the three color perceptual attributes of hue, lightness, and saturation.

The work described in this thesis is mainly to study the application of watersheds as a clustering techniques in two-dimensional color space. It should be noted that we are using color in a physical, rather than a purely psychological sense. The work concentrated on two main areas. First, the methods are described and the performance of the methods are illustrated. The possible application of this method in industrial usage using normalized color to eliminate the problem of shadow in color scenes is discussed.

Second, these methods are extended to segment the image into regions recursively. Different types of color images including man made color, and natural color have been used to illustrate the performance of the methods. Finally there is a brief discussion and a simple illustration showing how segmentation can be used in image compression and the application of pyramidal data structure in clustering for coarse segmentation.

The thesis consists of six main chapters. This chapter contains the outline of main

areas of image processing.

Chapter 2 provides a review of image segmentation methods. Both segmentations based on edges and regions are discussed. Finally, there is a brief discussion of the relationship between the methods.

Chapter 3 introduces some basic concepts in euclidean and digital mathematical morphology for image processing, which form a background to the methods used later, and also serves as a review of mathematical morphology.

Chapter 4 develops the color image segmentation algorithm using watersheds, and illustrates how the noise cleaning and edge detection can be implemented by using dilation and erosion, and also illustrates how the fragmentation caused by the shadow problem in color scenes can be partly eliminated using normalized color. Finally, there is discussion of application of this method to industrial usage.

Chapter 5 develops the recursive clustering algorithm using watersheds for color image segmentation. These methods have been applied to man made color images, and natural color images. Finally, there is an illustration showing how segmentation can be used in image compression by means of regional description and boundary description, and the application of pyramidal data structure in clustering for coarse segmentation.

Chapter 6 discusses methods which can provide some improvement to the segmentation. These methods include pre-processing of edge extraction and texture extraction, and post-processing of merging. Finally, there is a brief discussion in segmentation errors.

Chapter 7 provides the conclusions of the thesis, and suggests possible areas of future research that would improve the methods already developed.

Chapter 2

REVIEW OF IMAGE SEGMENTATION TECHNIQUES

2.1 Introduction

For various applications such as data compression discussed in section 5.4, pattern recognition, descriptive economy, or as a stage in the process of image understanding, it is often useful to segment pictures into regions. Region segmentation has been described by [Gonzalez & Wintz 87] as follows. Let R represent the entire image region. Segmentation may be viewed as a process that partitions R into n subregions, $R_1, R_2, ..., R_n$, such that

1. $\bigcup_{i=1}^{n} R_i = R$,
2. R_i is a connected region, $i = 1, 2, ..., n$,
3. $R_i \cap R_j = \emptyset$ for all i and j, and $i \neq j$,

4. $P(R_i)$ = TRUE for $i = 1, 2, ..., n$,

5. $P(R_i \cup R_j)$ = FALSE for $i \neq j$, where R_i and R_j are adjacent.

where $P(R_i)$ is a logical predicate defined over the points in set R_i, and $\emptyset$ is the null set.

The first condition indicates that every pixel must be in a region. This means that segmentation should not terminate until every point has been processed. The second condition requires that points in a region must be connected. The third condition indicates that the regions must be disjoint. The fourth condition determines what kind of properties the segmented regions should have. One simple example is uniform gray levels in which $P(R_i)$ = TRUE if all pixels in R_i have the same intensity. The final condition indicates that adjacent regions R_i and R_j are different in the sense of predicate P.

Segmentation is the process which divides the image into parts such that each segment has a homogeneous property such as intensity, color or texture etc. Interesting features can then be extracted for subsequent processing, such as description, recognition and identification of complete objects. For example in an earth resource study from satellite images; the problem may be to find homogeneous regions with respect to some characteristics such as gray level, color or texture, and to classify them into various land-use categories such as city, forest, water, and agricultural fields. To achieve this result the image is partitioned such that:

1. Regions of an segmented image should be uniform and homogeneous (i.e. pixels belonging to the same region must have approximately the same gray level, same color or same texture).

2. Adjacent regions of a segmented image should have significantly different values with respect to the characteristic on which each is uniform (i.e. pixels belonging

to adjacent regions must have significantly different gray level, different color or different texture).

It is, sometimes, not easy to achieve both properties because using strictly uniform and homogeneous regions will break the image into many small pieces. Further, insisting that adjacent regions should have large differences in values can cause regions to merge. As a result boundaries are lost and the objects of interest may not be found. Segmentation has generally been done under the assumption that:

1. The area representing an object is more or less uniform in its local properties: gray level, color, texture, and so on.

2. There is a detectable discontinuity in local properties between the areas representing two distinguishable objects.

These two assumptions result in two methods:

1. Segmentation based on areas (regions): the process of region finding is the partitioning of an image into its disjoint subsets corresponding to distinguishable regions in a scene.

2. Segmentation based on edges: in this case the edge is the boundary between two adjacent regions. Generally, edge finding is a process of detecting the parts of an image where there are abrupt changes in local properties, and it is frequently difficult to use such edges to define regions.

2.2 Review of Image Segmentation Techniques Based on Regions

Image segmentation techniques based on regions can be classified into two categories: measurement space guided spatial clustering, and region extraction. The former methods do not make use of spatial information on the image to form regions, but the latter methods do use that information [Haralick & Shapiro 85].

Since the measurement space guided clustering methods are based on the assumption that the similarity of measurement values of each class of a segmented region form a 'mode' in the measurement space, these methods are relatively immune to noise but will fail if this assumption is not true.

Measurement space guided spatial clustering methods work by using a measurement space clustering process to define a partition in measurement space. Each pixel on the image is assigned the label of the 'mode' in the measurement space partition to which it belongs. The segments are defined as the connected components of the pixels having the same label. The accuracy of the measurement space clustering processing depends directly on how well the objects of interest on the image separate into distinct measurement space clusters.

The simplest measurement space clustering is 'global thresholding', which can be accomplished by determining the valleys in the histogram and declaring the clustering to be the intervals of values between valleys. Since each uniform region contains many pixels of similar value, peaks in the histogram (called 'modes') are considered to correspond to such regions. If the histogram contains multiple peaks, the region is regarded as being composed of multiple uniform regions, and can be split by using 'multilevel thresholding'

as follows: a pixel whose value is in the i th interval is labeled with i and the segment it belongs to is one of the connected components of all pixels whose label is i.

In the case where the objects to be extracted are very small and thinly scattered, the picture consists almost entirely of background, and the objects may not produce detectable peaks on its histogram. This problem can be solved by using 'local thresholding' in which the whole image is divided into a number of small sub-images. Each sub-image is then used to form an histogram. If the sub-image contains objects, the corresponding histogram should have peaks. Therefore local thresholds can be selected for these sub-images to separate the object from the background.

The technique is extended to make use of multiple histograms to do recursive segmentation in the following way: the segmented regions are considered as new images and the process of histogramming, peak selection, and clustering is repeated until no new peaks can be found or regions become too small. The effective implementation of this method are described by [Tomita et al 73] for texture images, and by [Ohlander et al 78] for color images.

However, the histogramming technique is not suitable for images that include non-uniform regions, where the histogram contains no prominent peaks because pixel values within each region are not uniform. Nonuniformity inside the image would result in a very large number of small regions. Since the clustering is done in measurement space and no spatial information inherent on a image is taken into account, the resulting boundaries are very noisy and busy [Haralick & Shapiro 85].

Region extraction methods make use of spatial information to form regions. They can be classified as: region merging, region splitting, and region splitting and merging.

Region merging involves finding small groups of pixels (which can be individual pixels

or collection of adjacent pixels) with similar properties such as gray level, color, texture, then merging adjacent similar regions until no adjacent regions are sufficiently similar to be merged. The typical region merging algorithm performs a calculation of the similarity for each pair of adjacent regions, and the best pair are merged together. One example criterion of similarity of two adjacent regions is the difference of the average gray levels. If the seed region used is a single pixel and the segmented regions should be highly homogeneous, the worst case is that each segmented region be a single pixel. On the other hand if we insist adjacent regions should have large differences in values, the worst case is that only one segmented region, consisting of the whole image, can be found. However, region merging techniques work very well in the ideal case (i.e. images in which regions are basically homogeneous with small variations in the properties used such as intensity).

A way to extract the seed regions is to use edges [Shirai 87]. After applying an edge detector to the image, edges are extracted in the gradient image by thresholding. Seed regions are determined as those enclosed by edges. Edges with open ends are neglected. In order to obtain many seed regions, the threshold for edge extraction must be low enough so that edges with low contrast may be detected. A similar approach used to find the seed points has also been used by [Levine 80].

Region merging can be implemented by linking pixels together in graphs. Regions in an image can be represented in the form of graphs, in which the nodes represent regions, and two regions are joined by an arc if they are adjacent. The properties of each region (average gray level, color, etc.) are associated with each node. Similarly the properties of each pair of regions (length and strength of common border, etc.) are associated with each arc. The region merges can then be carried out by contracting the graph. The properties of a new merged region are recomputed whenever two nodes are merged. [Asano & Yokoya

81] and [Suk & Cho 83] used a similar approach to do the region merging.

One of the disadvantages of the region merging process is its inherently sequential nature. The regions produced depend on the order in which regions are merged together. In order to solve these problem, the methods proposed by [Burt et al 81][Hong & Rosenfeld 84] adopt a pyramidal representation of the image, in which the image is represented at a number of different spatial resolutions, and a heuristically derived 'relation' type of process is used to merge the regions. The method takes advantage of the pyramidal image description to combine local and global information in guiding the segmentation process.

Region splitting is the opposite approach to merging for segmentation. Starting with a large region such as the entire image, split it into multiple regions. Continue the splitting process until no further splitting is possible. The key to this method is a criterion for deciding how a region should be split. [Klinger & Dyer 76] proposed regular decomposition for image segmentation, using a top-down recursive partitioning of picture area into successively finer quadrants to obtain resulting tree structures for images.

The boundaries produced by this region splitting technique tend to be squarish and slightly artificial because regions are successively divided into quarters. The final partition may contain adjacent regions with identical properties. This may be remedied by allowing merging, as well as splitting. The general idea is to start with a given initial partition such as squares of a fixed size, $R_1, R_2, ..., R_n$, merge adjacent regions if the resulting new region is sufficiently homogeneous i.e., $P(R_i \cup R_j) = \text{TRUE}$, and then split a region if it is not homogeneous enough i.e., $P(R_i) = \text{FALSE}$. The process will stop if no further merging or splitting is possible.

A good method which can be used to represent the split and merge segmentation is the segmentation tree, where vertices correspond to regions and branches denote the

relationship between these connected regions. The splitting and merging of regions are expressed in terms of moving down and up the branches. A good example using split and merge method can be found in [Horowitz & Pavlidis 74,76]. They used a quad-tree for their segmentation algorithm in which each region is divided into four quarters in a recursive manner.

2.3 Review of Image Segmentation Techniques Based on Edges

Edge detection techniques, which are based on the concept of discontinuity, work by detecting boundaries between homogeneous regions of some properties such as intensity. Such techniques are well known for their limitations which are:

1. Edge detection by differentiation is easily affected by high frequency noise, especially when a small mask is used (i.e. an edge is characterized as a local intensity discontinuity).

2. Sometimes edges detected are not the boundaries between what we wish to consider as regions. They may be caused by a shadow lying on a region.

3. Homogeneous regions with smooth changes at the boundaries may not be detected.

4. Closed boundaries of regions are difficult to find in practice, since edges detected often have gaps where the transitions between regions are not abrupt enough due to blur, noise, insufficient contrast etc. Therefore, an additional linking step such as hough transform or edge relaxation, must be used to obtain a representation of the connected edge segment.

5. After the linking step, the original segment may change. Consequentially, reprocessing of the properties of some region may be required (for example, average gray level, color).

[Davis 75] has written an excellent survey on different edge extraction techniques, and classifies them into two groups: parallel techniques, and sequential techniques.

Parallel edge detection techniques are based on the result of convolving a local mask about the pixel in the image. Since the decision made is not dependent on previous results, the edge detection operators can be applied simultaneously everywhere in the image. However, the result is quite susceptible to noise, since the computation is based on a small window. The Kirsch, Sobel, and Prewitt operators are examples of these techniques [Fu & Mui 81], and are based on a 3×3 pixel masks.

Another approach in parallel edge detection is boundary detection based on formal models of edges, in which a model is defined and are fitted to a sample region of the input image in an optimal way. The Griffith, Hueckel and Chow operators are examples of these techniques [Davis 75].

Edge detection algorithms are seriously affected by noise on the image because they do approximate differentiation operations, calculating the rate of change of pixel intensity. Thus, the noise-characteristic of an edge detector depends on its size. The larger the size of the detector, the more random noise will be averaged out. However, it is also more likely to overlap several edges or corners simultaneously and thus degrade the resolution capability. The effect of noise can also be reduced by pre-processing with a smoothing filter (i.e., low pass filter), but as a result the edge will be blurred and the sensitivity of the edge detector reduced.

[Marr & Hildreth 80] have suggested a compromise method using an operator consisting of a Laplacian of a Gaussian. The Gaussian serves as a smoothing function and the Laplacian gives a nondirectional derivative, in which the second derivative is zero at the edge but has a positive and negative peak on either side. Thus, edge detection is accomplished by locating zero crossings. High or low resolution of the operator can be obtained by using a sharp or wide Gaussian respectively. This operator is considered as a good model of the processing in the human visual system, and has been used extensively for work in stereo vision [Grimson 80].

The parallel edge detection techniques are not guaranteed to produce a set of closed connected curves that surround connected regions, because of noise, breaks in the boundary due to nonuniform illumination, and other effects that introduce spurious intensity discontinuities. Thus edge detection techniques are typically followed by edge linking procedures designed to assemble edge pixels into a meaningful set of region boundaries. One possible approach is the Hough transform [Hough 62]. In this method, the edges are transformed to another space, called the Hough space, with the property that the desired groups of edges cluster in the transform space.

Sequential edge detection techniques, in which the result at a point is dependent on the results of the operator at previously examined points, are used to force some continuity between edge points. There are a number of sequential techniques such as using heuristic search, dynamic programming, and guided edge detection [Davis 75]. The major components of a sequential edge detection techniques are:

1. The picking of a good initial point: since the result will depend on the present input and the previous result, the performance of the entire procedure will depend upon the choice of a good starting point.

2. The dependence structure: how do the results obtained at previously examined points affect both the choice of the next point to be examined and the result at the next point?

3. A termination criterion: there must be a way for the procedure to determine that it is finished.

2.4 Review of Region and Edge Extraction Using Color

The term 'color' is often used where it is really multispectral signals which are being used. Frequently, color television techniques are applied as these are easily available. See section 4.2 for a further discussion of this.

[Ohlander et al 78] used nine one-dimensional histograms of color features: the red, green, and blue color components; the intensity, saturation, and hue components; and the National Television Systems Committee (N.T.S.C.) Y,I,Q components. Among the nine histograms, one is selected which contains the most prominent peaks and valleys. The regions are then thresholded. Every time new regions are produced, the nine histogram are calculated and a decision is made on the basis of the result of those calculations. This procedure is repeated until no threshold is suggested by any histogram. Similar methods, which integrated edge information, have been proposed by [Milgram & Kahl 79] [Milgram & Herman 80].

[Schacter et al 76] used '3-dimensional histogram' of (RGB) color features to do the clustering. The 3-dimensional histogram is stored as a binary tree by using the triples of color values as key, and the information is the number of the points with this key value. To construct the tree, the triple of color values at each scene point is used to search the tree for this key. If the key is found, the point count at that tree node is incremented by 1,

and if not, a node with that key is added to the tree with its count set to 1. Clusters are detected if the number of the points exceeds some threshold and if they lie within some distance from each other. This method is computationally expensive, since it searches for the optimal cluster locations and the optimal number of clusters by repeated iterations over the data.

A similar technique for clustering was employed by [Sarabi & Aggarwal 81], where an interactive system that use 3-dimensional clustering in the normalized color space (X,Y,I) representation of the scene solves segmentation when there is no prior knowledge about the color space characteristics of the scene. When particular color space characteristics are known, the color space is projected onto the X-Y, X-I, or Y-I faces of the normalized color space as described in [Underwood & Aggarwal 77]. They determined the threshold interval to extract the region by manual adjusting for their interactive system.

[Khotanzad & Bouarfa 90] described a non-parametric clustering algorithm, which is totally automatic and is easily parallelizable, and its application to unsupervised image segmentation. Image segmentation is performed by clustering features extracted from small local areas of the image. The clusters are found by mode analysis of the multi-dimensional histogram of the considered vectors through a non-iterative peak-climbing approach.

[Celenk 90] described a clustering technique and its use in segmenting the color image of natural scenes. The proposed method operates in the 1976 CIE (L,a,b) uniform color coordinate system using cylindrical coordinates. It detects image clusters in the cylindrical color space by means of estimating their distributions in some well-defined decision volumes of the constant lightness and constant chromaticity loci.

[Chang et al 87] have proposed a new threshold selection algorithm to determine the

threshold interval that extracts the single cluster nearest to a specified feature vector in multi-dimensional feature space. This algorithm has been applied to color images for extracting a uniform color region of specified R,G,B intensities, and they found that the extracted color regions are well matched with the specified real color region.

[Funakubo 84] reported that color features which consists of three gray level values (i.e. red, green, and blue) do not provide sufficient information for extracting regions from such a complex image as biomedical tissue image. In such the case textural feature, which is determined by the spatial distribution in respect to color information of pixels, might be used to improve the segmentation effectively.

There are few reports of using human color perception model for segmentation. Two examples are the research done by Tominaga and Garbay.

[Tominaga 86,87,88] described a method for segmenting a color image into uniform color regions by means of the three color perceptual attributes of hue, lightness, and saturation. The Munsell color system, which is used as the color space for specifying human color perception, provides a perceptually uniform color space defined in the three attributes called Munsell Hue, Value, and Chroma. They are determined from human perceptual experience of object colors. Firstly, a mapping method is used to transform the observed color signals into the color space [Tominaga 83,84], so that the perceptual attributes of a color image can be predicated quantitatively. The histograms of the three attributes are then analysed for image segmentation by recursive thresholding method.

Another human color perception model was proposed by [Faugeras 79]. The basic feature of the model is a linear transformation of the R,G,B log-images into one achromatic image and two chromatic images. Each color is then characterized in the perceptual space by its luminance (L), saturation (S) and hue (H). Garbay used this model for segmentation

of bone marrow cells, in which a segmentation algorithm based on a linear discriminant analysis is used. It involves the partitioning of the space generated by the vectors (L,S and H) into four clusters corresponding to the background, the erythrocytes, the cell cytoplasm and the cell nucleus, considered as four classes of objects. This approach was shown to increase the efficiency of the segmentation, feature extraction and classification of bone marrow cells [Garbay 81,82][Chassery & Garbay 83,84].

[Ito 75,76,80] utilized color information in an industrial inspection system for IC mask patterns. Patterns of IC masks were illuminated by red, green, and blue light, respectively. Defects of these mask pattern could be identified by the optical composition of the patterns under illumination of the primary color light source.

Two color coordinate systems for color image recognition for industrial automation have been evaluated by [Asano et al 86]. The two color systems are RGB and Chrominance signal (Y,R-Y,B-Y). In both systems the color image are transformed into hue and saturation image, and the luminance signal is not used for recognition. They find that both systems have similar hue resolution; however the hue/saturation transformation in chrominance signal system is faster than that in RGB system.

A similar technique, which uses the color system, NTSC YIQ, for color image analysis, has been discussed by [Kelley & Faedo 85]. In their experiments, on detecting resistor color bands, the results show that phase-magnitude representation of the IQ-plane chrominance data leads to computationally efficient scalar segmentation algorithms. Saturated colors can be distinguished using phase data alone while nonsaturated colors require chrominance and intensity data.

[Aus et al 83] suggested that the luminance and chrominance signal obtained from an ordinary color TV camera might be useful in scene segmentation. After checking the

white balance and registration of the three tubes of a color TV camera, a color scene is captured and stored as three gray level images representing the red, green and blue component image. The P.A.L. Y,U,V at each pixel in the image are calculated using a software algorithm, then the user can interactively select color phase, color amplitude, and luminance thresholds to segment the scene.

[Keil 83] used a chromakeyer to convert the hue content of a N.T.S.C. Y,I,Q, color television image into a standard monochrome television image. The chromakeyer is used to select a desired hue, which represents the object of interest. All occurrences of this hue in the color image are enhanced with a high gray level value and its complementary color is suppressed with a low gray level value. All other colors become various shades of gray depending on their relative positions in the IQ-plane. The particular characteristic of this system is that not all of the information which is available in the color image is used.

Another industrial inspection system using color spectral signatures for color recognition was reported by [Berry 87]. Firstly, experiments are conducted to determine the spectral features that are invariant over a range of lighting conditions. Having determined those features, an automatic color recognition scheme which makes use of both color and gray-scale information is developed for spray paint can tops. Input to the system is a color image and output is the dimensions of the center, radius in screen coordinate and color of the spray paint caps.

The performance of segmentation depends not only on its segmentation algorithm, but greatly on color features used in its segmentation processes. [Ohta et al 80] conducted a systematic experiment of segmentation in natural scenes using different color features, and suggested three effective color features in terms of a linear transformation of the R,G,B color space. The three features are $(R+G+B)/3$, $(R-B)/2$, and $(2G-R-B)/4$, and

these are used to find the threshold instead of R,G,and B.

The edges in the color image are found to be largely contained in the edges in the intensity image. This implies that most of the information of interest is embedded in the intensity image. On the other hand, if intensity edges are absent due to low contrast, color edges can resolve the problem of finding edges. Besides, the color edges can be used to determine the edges with higher confidence particularly for edges that occur in both the intensity image and color image [Nevatia 76].

The simplest scheme is perhaps to compute edges in the three color components separately and determine an edge in the color image if certain relations between edges in individual components are satisfied, e.g., we can take their RMS, or the sum or maximum of their absolute values [Rosenfeld & Kak 82]. For example, let R(i,j), G(i,j), and B(i,j) be the red, green, and blue components. Then the RMS of the difference between points (i,j) and (i-1,j) are

$$[(R(i,j)-R(i-1,j))^2+(G(i,j)-G(i-1,j))^2+(B(i,j)-B(i-1,j))^2]^{1/2}$$

which is just the Euclidean distance between the color vectors

$$[R(i,j),G(i,j),B(i,j)] \text{ and } [R(i-1,j),G(i-1,j),B(i-1,j)].$$

A similar technique has been used by [Robinsion 76], in which the maximum value of the gradient of the three color components at a point in an image are calculated, and then combined to form a color edge. This is essentially a gray level edge technique, and thus the result is dependent on the choice of suitable thresholds for each gradient color component image. One of the biggest problem is selection an optimal threshold value automatically.

[Nevatia 76,77] developed a color edge detector based on a generalization of an edge operator developed by [Hueckel 71,73] for a single gray level images. The Hueckel operator determines the presence of an edge in a circular neighbourhood by fitting an optimal step

to the input signal. An ideal edge step is defined to have a step or line profile in brightness, this profile being constant along a straight line in the signal window. Nevatia extended this concept of edge to a color edge by applying the operator to the three components separately. Presence of an edge in the color image may now be based on the relationship between the edges in the three components.

2.5 The Relationship Between Region Based and Edge Based Segmentation Methods

Region based and edge based segmentation methods are closely related, since region based methods attempt to locate homogeneous regions in the image and edge based methods attempt to locate boundaries between regions in the image. Naturally, borders of regions may be identified as edges. So it is reasonable to assume that relating these two processes together can improve an segmentation results which depend on just one method. However, it is hard to establish the relationship between these two methods.

On the other hand, if the relationship between these two methods can be established, these two processes can be unified. One possible method, which can establish such relationship using region and region boundaries, is graph theory [Morris et al 86]. It uses the vertices to represent the regions. Regions are connected by links if they share an edge (i.e. adjacent). Using graph duality, we can derive the relationship between vertices and links in a graph and the spaces that are formed between the links. An recent example using integration of both methods, region growing and edge detection, is described in [Pavlidis & Liow 90].

However, it is hard to establish such relationship for the region based methods which use the clustering techniques. Since the clustering is done in feature space, there is no

spatial information from the image. Besides, the contradictory regions and edges produced are not easy to solve, unless we have some prior knowledge about the type of image. It also explains why most papers consider these two methods in almost total isolation from each other.

2.6 Summary

The aim of the present chapter is to review the general segmentation techniques based on both region and edge. The wide variety of methods have been reviewed indicates that many different approaches are possible. However, one major advantage of using region based methods over edge based methods is the former always produce closed regions and those regions can constitute the whole image, but the set of all edges obtained using the latter usually does not work.

So far, there is no standard approach to image segmentation and also no theory of it, and the methods used are strongly problem dependent. However, it seems further improve can be obtained, if we can combine edge detection and region segmentation with semantic information to perform image segmentation. A good survey on image segmentation methods can be found in [Davis 75] [Zucker 76][Fu & Mui 81][Haralick & Shapiro 85].

Chapter 3

INTRODUCTION TO MATHEMATICAL MORPHOLOGY FOR IMAGE PROCESSING

3.1 Introduction

Morphology refers to the study of form and structure. Mathematical morphology provides an approach to the processing of images, where the images being analysed are considered as a set of points and the operations come from set theory [Matheron 75][Serra 80,88][Haralick et al 87]. Morphological operations can be employed for many purposes, including edge detection, segmentation, and enhancement of images. Indeed, one of the most useful areas of morphological analysis is the generation of feature parameters for use in artificial intelligence schemes, since the morphological operations manipulated on pictorial content

rather than pixel states can rigorously quantify many aspects of the shape or geometrical structure of signals in a way that agrees with human intuition and perception [Maragos & Schafer 90]. In this chapter, we will discuss euclidean (or analog) and digital morphological operations. In order to define mathematical morphology, we first require some background definitions in usual set-theoretic operation of union and intersection.

1. Let $P(E)$ (the powerset of the set E) be considered as a complete lattice. For $X_i \in P(E)$, i = 1, 2, 3,..., the least upper bound of X_i is their union $\cup X_i$ and the greatest lower bound of X_i is their intersection $\cap X_i$, both of which belong to $P(E)$ (figure(3.1)).

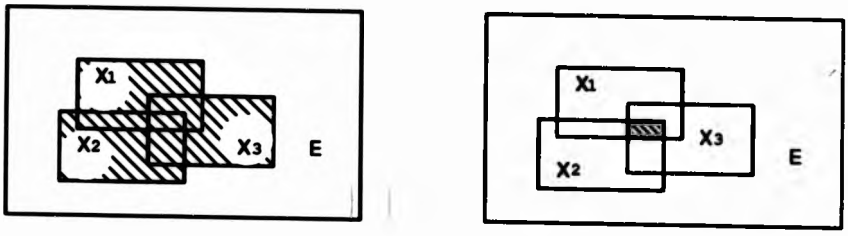


Figure 3.1: The block diagram for $LUB(X_i)$ and $GLB(X_i)$, where $i = 1, 2, 3$.

2. The lattice $P(E)$ is distributive since union and intersection are distributive, i.e.,

$$X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z) \qquad \forall X, Y, Z \in P(E)$$

3. The lattice $P(E)$ is complemented, i.e.,

$$X \cup X^c = E \quad \text{and} \quad X \cap X^c = \emptyset$$

where X^c is the complement of the set X and $\emptyset$ is the empty set (figure(3.2)).

4. Given two sets X and $Y \in P(E)$, their set difference X/Y can be derived from intersection and the complement as follows :

$X/Y = X \cap Y^c$

X/Y is the part of X which does not belong to Y (figure(3.3)).

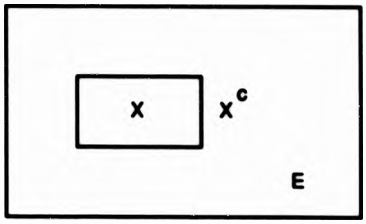


Figure 3.2: The block diagram for $X \cup X^c = E$ and $X \cap X^c = \emptyset$.

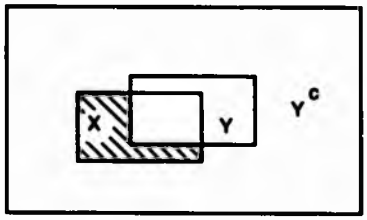


Figure 3.3: The block diagram for X/Y.

3.2 Euclidean Morphological Operations

This section discusses some fundamental morphological operations in euclidean space, including two important techniques of dilation and erosion. These techniques are then used to explain the closing and opening operations, and have also been used to illustrate edge detection. Although the actual implementation of these operators will be in digital setting, the euclidean model is essential to the development of an understanding how the operators function in both theory and application.

3.2.1 Primitive Euclidean Morphological Operators

1. Structuring element

   With each point x of the euclidean plane R^2 in which we work, we associated a set A(x) called a structuring element (i.e., a set of pixels constituting a specific shape such as a line or square).

2. Reflection

   Given an image (subset) A in R^2, the reflection of A is -A, which is simply A rotated 180^0 around the origin (figure(3.4)).

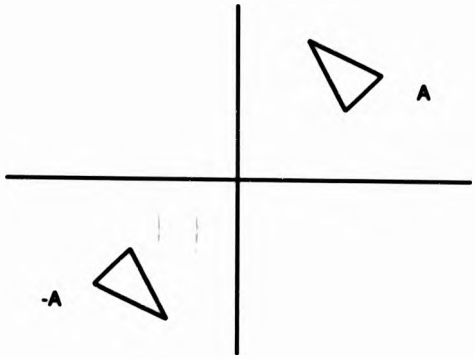


Figure 3.4: Illustration of reflection.

3. Translation

   Given an image (subset) A in R^2, the translation of A by the point x in R^2 is defined by

$$A + x = \{a + x : a \in A\}$$

   where the plus sign inside the set notation refers to vector addition (figure(3.5)). We can write $x + A$ interchangeably with $A + x$ because vector addition is commutative.

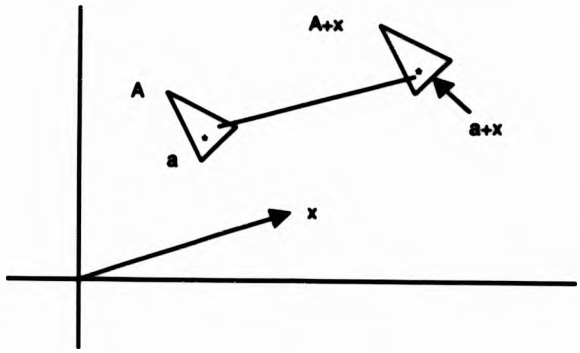


Figure 3.5: Illustration of translation.

4. Minkowski addition

Given two images (subsets) A and B in R^2, we define the Minkowski sum as

$$A \oplus B = \bigcup_{b \in B} (A + b)$$

In this operation, A is translated by every element of B and then the union is taken (figure(3.6)).

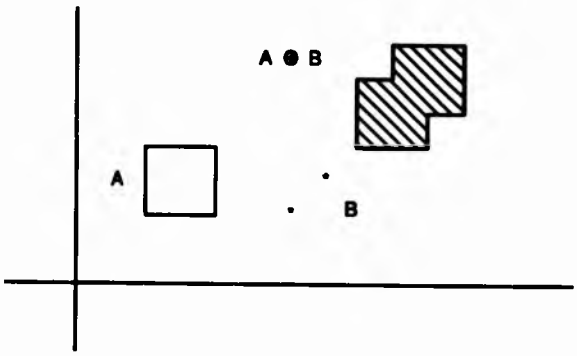


Figure 3.6: Illustration of Minkowski addition.

5. Minkowski subtraction

Given two images (subsets) A and B in R^2, we define the Minkowski difference as

$$A \ominus B = \bigcap_{b \in B} (A + b)$$

In this operation, A is translated by every element of B and then the intersection is taken (figure(3.7)).

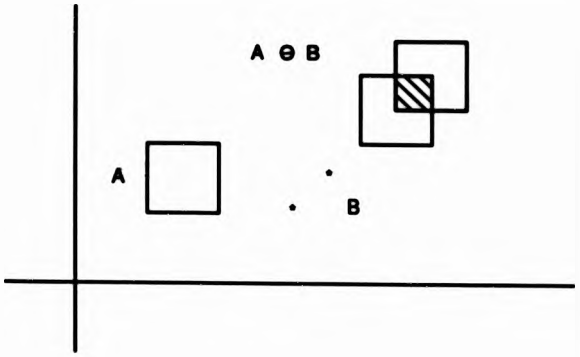


Figure 3.7: Illustration of Minkowski subtraction.

6. Erosion

We define erosion of A by B to be $E(A, B) = A \ominus (-B)$, where -B is simply B rotated 180^0 around the origin. If B=-B, the erosion is equal to Minkowski subtraction. When A is eroded by B, the latter is called a structuring element. Eroding an image by a structuring element B has the effect of 'shrinking' the image in a manner determined by B (figure(3.8)).

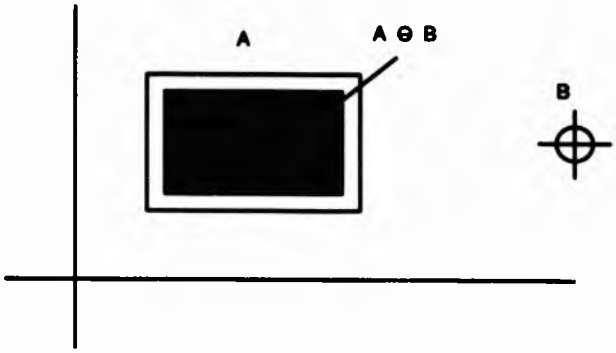


Figure 3.8: Illustration of the erosion as shrinking.

7. Dilation

Corresponding to the erosion operation is the operation of dilation, which is defined simply as Minkowski addition:

$$D(A, B) = A \oplus B$$

When A is dilated by B, the latter is called a structuring element. Dilating an image by a structuring element B has the effect of 'expanding' an image in a manner determined by B (figure(3.9)).

The dilation and erosion operations are duals because in the sense that the dilation of the foreground is equivalent to the erosion of the background (figure(3.10)).

8. Opening

In mathematical morphology, the process of eroding and then dilating by the same structuring element is called an opening.

$$O(A, B) = [A \ominus (-B)] \oplus B$$

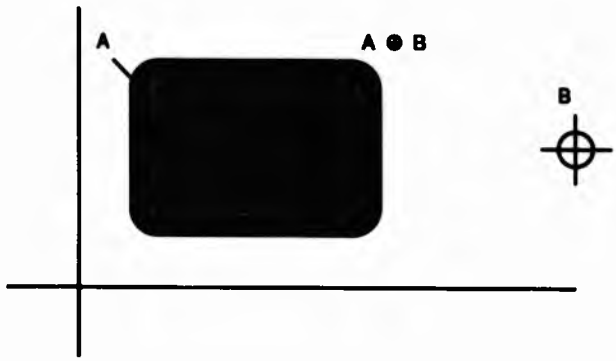


Figure 3.9: Illustration of the dilation as expansion.

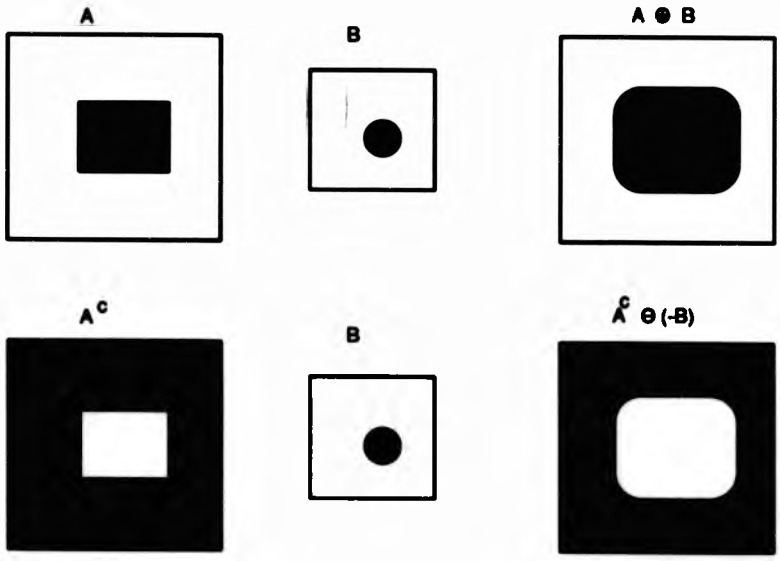


Figure 3.10: Illustration of the duality of dilation and erosion, $A \oplus B = (A^c \ominus (-B))^c$.

The opening of A by B is the union of all translations of B that can be included in A, or

$$O(A,B) = \bigcup\{B + x : B + x \subset A\}$$

which allows us to determine where the given structuring element B can fit as it 'slides around' within A. It will remove all the regions that are too small to contain the structuring element in the image being opened. If a disk-shaped structured element is used, all the regions smaller than the disk will be eliminated. This forms a filter that suppress the spatial details (figure(3.11)).

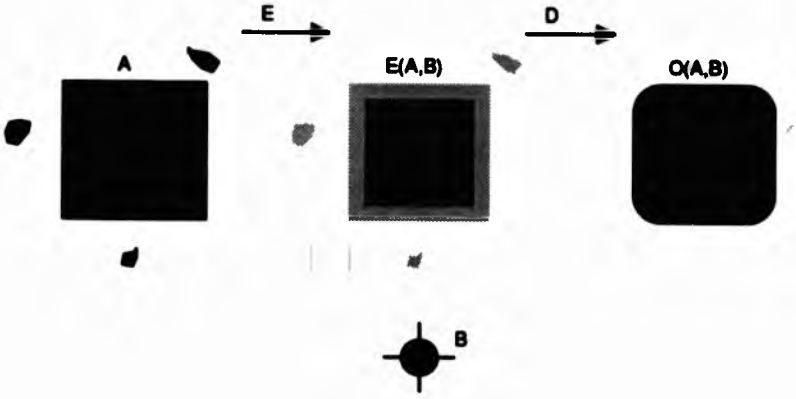


Figure 3.11: Illustration of opening.

9. Closing

In mathematical morphology, the process of dilating and then eroding by the same structuring elements is called a closing. This fills in holes smaller than the structuring element (figure(3.12)).

$$C(A,B) = [A \oplus (-B)] \ominus B$$

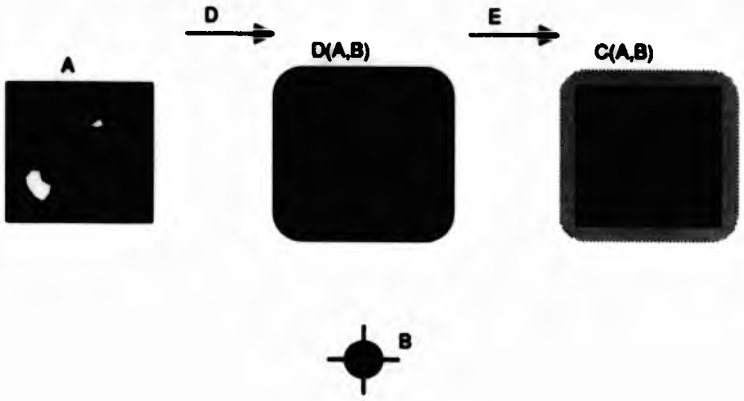


Figure 3.12: Illustration of closing.

10. Edge Detection

Dilation and erosion can also be used to detect the edges of an image. Consider an image A and a small symmetric structuring element B, then the set difference $(A \oplus B)/A$ gives the external edge of image A. The set difference $A/(A \ominus B)$ gives the internal edge of image A. By adding both the operations, new edge of image A can be obtained that treat more symmetrically the image and its background. However, the thickness of the edge is doubled. In terms of algebraic difference, the edge of image A can be defined as:

(a) External edge of $A = D(A, B) - A$

(b) Internal edge of $A = A - E(A, B)$

(c) True edge of A = External edge + Internal edge = $D(A, B) - E(A, B)$

The thickness of the edge of image A results from the thickness of B. Figure(3.13) illustrates the edge detection using dilation and erosion.

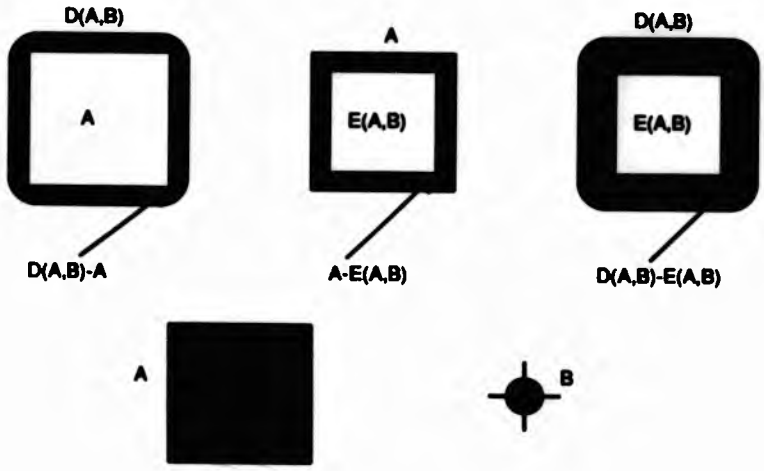


Figure 3.13: Illustration of edge detection using dilation and erosion.

3.2.2 Clustering by Watersheds

Clustering can be defined as dividing a set of data samples into a number of subsets (i.e. classes) such that all the members of one subset are similar enough and differ significantly from the members of the other subsets. Thus clusters can be identified by defining cluster centers which try to minimize the difference within a cluster of points, and try to maximize the difference between the centers of the clusters.

Normally there is no a priori knowledge of the number of subsets. Therefore, clustering can also be considered as a form of unsupervised classification and consists of determining both the number of clusters and the subset membership of the data samples [Hall 79]. That means the partitioning or grouping decisions are to be totally dependent on the data samples themselves. So the choice of the feature space, which the data samples are derived from, might be the major and most difficult problem in clustering, as the performance relies almost totally upon the feature space for each class of object being distinct. A

more detailed discussion in how the feature space affects the clustering can be found in section 4.1. It also explains why clustering is a classic method used in partial segmentation problems, since the image being segmented can be anything (such as landscape or urban estate) and therefore there is no a priori knowledge of the classes.

In this section we will discuss what watersheds are and then illustrate how this concept can be used as a clustering technique in two-dimensional data. Let us imagine that it rains over a geographical surface. The water streams down, reaches a minimum height and stops there. For each minimum, a set of all the points from which the water may come, is called a catchment basin. Several catchment basins may overlap and their common points form the watersheds as shown in figure(3.14).

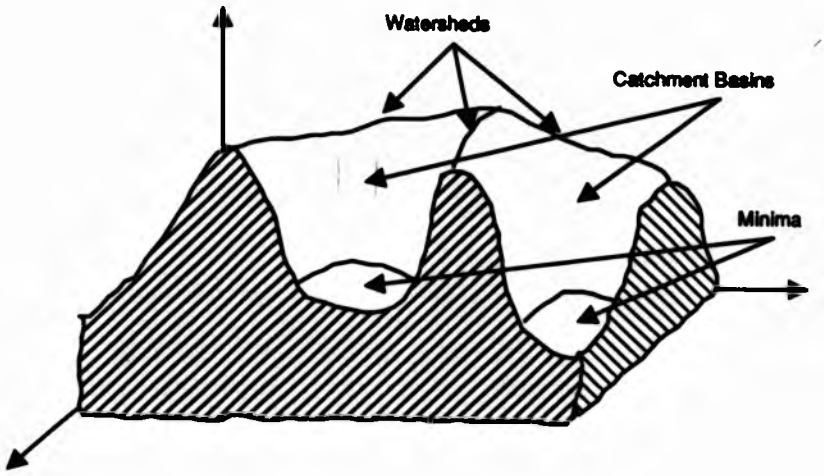


Figure 3.14: Watersheds and catchment basins

The watersheds of a continuous function from $R \times R$ to R intuitively correspond to the intersection of geographical watersheds [Beucher 82][Lantuejoul 82]. For some such functions, they form a way of dividing the functions domain into disjoint subsets: two

points of the domain are in the same subset if their function images belong to the same catchment basin. The watershed concept can equally be applied to functions mapping a two-dimensional discrete space into R. One application of watersheds was proposed in [Watson 87]. They were used to provide a technique for classifying satellite images which made no a priori assumptions about the nature of the classes into which the image would be divided. Thus an unbiased aid to the interpretation of an image can be obtained.

One can consider the geographical surface formed by a two-dimensional histogram in an inverted position so that the minima and maxima are interchanged. Their minima, catchment basins and watersheds can then be located by using the following method:

1. Locate and label the peaks in the two-dimensional histogram. The maximum detection algorithm consists of detecting the location at which the count is higher than elsewhere in the 3×3 square neighbourhoods. When peaks consist of more than one point (adjacent locations have the same count), these points will be considered as one single peak and marked with the same label (figure(3.15)). It is necessary to use 8-connected neighbourhoods to detect the peak, since the 4-connected neighbourhoods detection in which only the horizontal and vertical neighbour are examined will sometimes lead to false indications of peaks in the vicinity of true peaks (figure(3.15)).

2. Grow the region around each peak by constantly descending from the peak until whole two-dimensional histogram has been labelled.

Adjacent mountains meet in their common valleys (i.e. watersheds in the geographical surface), and are thus separated from each other. Each catchment basin region can be labelled, so that the whole histogram can be segmented, and the labels are used as a LUT (lookup table) for classifying the image. The motivation is using similarity in property (in

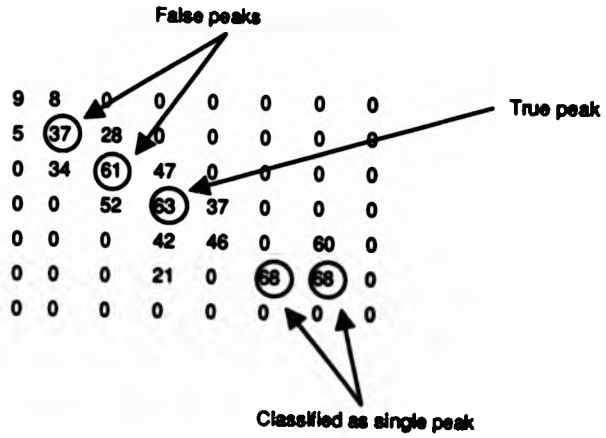


Figure 3.15: False indication of a peak when the peak detection is 4-connected.

this case color), without reference to distance in initial image, for segmentation.

The method above is illustrated in figure(3.16). Figure(3.16a) and (3.16b) are the two input bands: the R and G images (i.e. the image produced by using a red and green filter in front of a monochrome camera). Using these two bands, we may compute a two-dimensional table H(v,u) giving the number of pixels having value v in the R-image and value u in the G-image. This two-dimensional histogram, (sometimes called a scatterplot), of the two bands is shown in figure(3.16c). The clustering method consists of first identifying local peaks and iteratively connecting to these peaks any points which are lower than and are 8-connected to these peaks. If a point has the same distance from two or more peaks then it is assigned to the group whose peak has the maximal value. This process is shown in figures(3.16d) and (3.16e). The final two-dimensional histogram, figure(3.16f), consists of two disjoint regions. Each region is labelled, and hence can be used as a LUT to classify the original color image by each pair of R and G values suppling the coordinates within the table.

```
1 1 1 1 1 1 4
2 2 2 2 2 2 4
2 2 2 2 2 3 4
3 3 3 3 3 3 4
3 3 3 3 3 3 4
3 3 4 4 4 4 4
3 5 5 5 5 5 5
```

(a). The red component image.

```
1 2 3 6 7 8 2
1 1 2 2 3 4 2
5 6 7 8 9 2 2
2 2 3 3 4 5 3
6 7 7 7 8 8 4
8 9 7 7 6 6 5
2 2 2 4 5 6 7
```

(b). The green component image.

G-image

R-image	0	1	2	3	4	5	6	7	8	9	10
0	.	.	.	.	.	.	.	.	.	.	.
1	.	1	1	1	.	.	1	1	1	.	.
2	.	2	2	1	1	1	1	1	1	1	.
3	.	.	4	2	1	1	1	3	3	1	.
4	.	.	3	1	1	1	2	2	.	.	.
5	.	.	2	.	1	1	1	1	.	.	.
6	.	.	.	.	.	.	.	.	.	.	.

(c). The 2-dimensional histogram of R and G component image.

```
. . . . . . . . . . .
. 1 1 1 . . 1 1 1 . .
. 2 2 1 1 1 1 1 1 .
. . a 2 1 1 1 b b 1 .
. . 3 1 1 1 2 2 . . .
. . 2 . 1 1 1 1 . . .
. . . . . . . . . . .
```

(d). Two peaks, '4' and '3', are found and are represented by labels a and b.

```
. . . . . . . . . . .
. 1 1 1 . . 1 1 1 . .
. a a a 1 1 b b b b .
. . a a 1 1 b b b b .
. . a a 1 1 b b . . .
. . 2 . 1 1 1 1 . . .
. . . . . . . . . . .
```

(e). After one pass of the region growing some of the pixels are connected to the existing groups.

```
. . . . . . . . . . .
. a a a . . b b b . .
. a a a a b b b b b .
. . a a a b b b b b .
. . a a a b b b . . .
. . a . a b b b . . .
. . . . . . . . . . .
```

(f). Repeating the region growing until no further additions can be made. The final results are two groups labelled with a and b.

Figue 3.16 : Illustration of clustering using watersheds.

3.3 Digital Morphological Operations

Digital morphological image processing is characterized by those algorithms that can be formed through the use of operators from the structural basis and the morphological basis. The processing power of morphological methods are limited by the nature of these basic operators. This section starts with a review of the bound matrix used to represent a digital image. This is followed by a discussion of basic structural operators, and a discussion of basic morphological operators. Lastly there is a discussion of the fundamental morphological operations based on the basic structural and morphological operators.

A digital image F can be defined as the function $F : D \rightarrow R$, and $D \subset Z \times Z$ where Z and R denote the sets of integers and real numbers[1]. D (which is used to denote a subset of the set $Z \times Z$) is called the domain of the image, and R is called the range of the image. A digital image can be represented using the bound matrix. [Giardina & Dougherty 88] described the bound matrix representation of an image as follows: assuming D is finite, then F can be represented as an $m \times n$ array.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}_{l,t}$$

where

1. (l,t) denotes the position of the upper left entry $a_{1,1}$ in the $Z \times Z$ grid.

[1]The pattern into which the image is divided is called its tessellation, a rectangular tessellation has been assumed here.

2. $a_{p,q}$ denotes the gray level value of F at the pixel (p,q) when the array is laid over the grid so that $a_{1,1}$ is over pixel (l,t).

3. $a_{p,q}$ is a star (*) if F is not defined at the pixel (p,q).

Since F has value $a_{1,1}$ at the pixel (l,t), all gray level values of F can be found by positioning the matrix at the specified (l,t) location and reading off the corresponding matrix values. The bound matrix representation of an image can be changed by employing extraneous columns or rows of stars. However, a minimal bound matrix is usually used to represent a digital image F with finite domain because it is convenient and space saving. The minimal bound matrix for F is simply the representation for which m and n are as small as possible.

3.3.1 Structural Operators

The operators, DOMAIN,RANGE, and CREATE, which will be referred to as the structural basis of digital image processing, are defined as follows:

1. DOMAIN

   The operator DOMAIN takes an image F as input and yields an array of ordered pairs that make up the domain of the image (figure(3.17)).

Figure 3.17: The block diagram for DOMAIN.

2. RANGE

   The operator RANGE takes an image F input and yields an array consisting of the gray values of the input image (figure(3.18)).

Figure 3.18: The block diagram for RANGE.

3. CREATE

The operator CREATE takes an array consisting of real numbers R and an array of ordered integer pairs D as inputs and outputs an image (figure (3.19)).

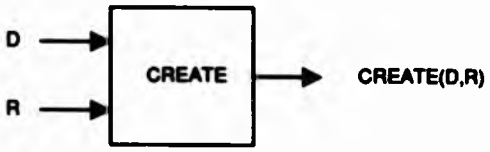


Figure 3.19: The block diagram for CREATE.

The following example will illustrate the operations of DOMAIN, RANGE, and CREATE. Let

$$F = \begin{bmatrix} 3 & 5 & * \\ 0 & 3 & 1 \end{bmatrix}_{1,1}$$

Then

$$DOMAIN(F) = (1,1),(1,0),(2,1),(2,0),(3,0)$$

$$RANGE(F) = 3,0,5,3,1$$

$$CREATE[(DOMAIN(F), RANGE(F)] = F$$

3.3.2 Primitive Digital Morphological Operators

The five elementary operators, EXTMAX, MIN,TRAN, NINETY, and COMP, which serve as the fundamental set of primitives for morphological image processing, are referred to as the morphological basis. After a brief discussion of these operators, they will be used to illustrate the four fundamental operations of dilation, erosion, opening, and closing in next section.

1. EXTMAX

The extended maximum operator EXTMAX compares two images in a pixelwise manner and outputs the maximum value at each pixel at which both images are defined. If both images are undefined at a specific pixel, then * is the output of EXTMAX. However, if only one image is undefined at a pixel while the other is defined, then the output of EXTMAX is the gray value of the defined image at the given pixel. The definition of EXTMAX can be extended to more than two input images by taking the maximum value over all input gray level values defined at the pixel, and outputting a * at pixels outside the union of the input image domains.

The extended maximum operator EXTMAX can be defined as follows:

$$[EXTMAX(F,G)](i,j) = \begin{cases} max[F(i,j), G(i,j)], & \text{if both F and G are defined at } (i,j) \\ F(i,j), & \text{if } F(i,j) \neq * \text{ and } G(i,j) = * \\ G(i,j), & \text{if } G(i,j) = * \text{ and } F(i,j) \neq * \\ *, & \text{if } F(i,j) = G(i,j) = *. \end{cases}$$

The domain of the output image is the union of the input domains.

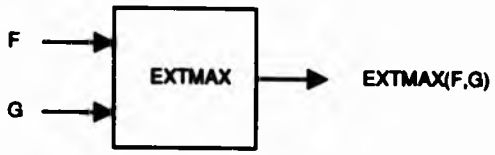


Figure 3.20: The block diagram for EXTMAX.

2. MIN

The minimum operator MIN, which is similar to EXTMAX in that a pixelwise comparison is taken, can be defined as follows:

$$[MIN(F,G)](i,j) = \begin{cases} min[F(i,j), G(i,j)], & \text{if both F and G are defined at } (i,j) \\ *, & \text{if either F or G is not defined at } (i,j). \end{cases}$$

The domain of the output image is the intersection of the input domains. The definition of MIN can be extended to more than two input images by taking the minimum of the gray level values on the intersection of the input domains and by being undefined (*) elsewhere.

Figure 3.21: The block diagram for MIN.

3. TRAN

The translator operator TRAN operates on the image F and two integers i and j, and produces an image which is identical to F but moved i pixels to the right and j pixels up (figure(3.22)). Thus TRAN leaves the gray level values of an input image intact while altering the domain of the image.

The translator can be defined pixelwise by

$$[TRAN(F;(i,j))](u,v) = F(u-i, v-j)$$

where i,j represent input, while u,v correspond to the coordinates of pixel undercon-sideration.

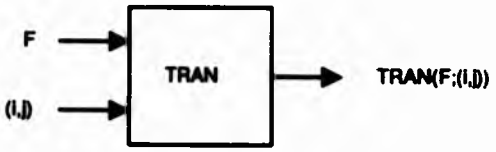


Figure 3.22: The block diagram for TRAN.

4. NINETY

The rotation operator NINETY rotates an input image 90^0 in the counterclockwise direction about the origin and is defined pixelwise by

$$[NINETY(F)](i,j) = F(j,-i)$$

Figure 3.23: The block diagram for NINETY.

In morphological analysis, the double application of NINETY, plays a key role since it yields a reflection of the image through the origin. A general image can be rotated by 180^0 by means of two successive application of NINETY, we can call this operation $NINETY^2$ and its block diagram is given by

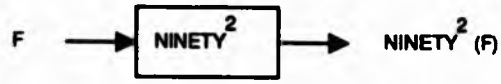


Figure 3.24: The block diagram for $NINETY^2$.

Similarly, a 270^0 rotation is equal to $NINETY^3$, NINETY applied three times in succession. The respective block diagram for $NINETY^3$ is given by

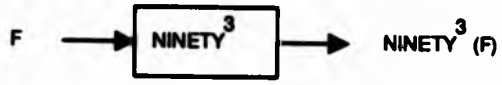


Figure 3.25: The block diagram for NINETY^3.

5. COMP

Let image S be a binary image which possesses two values: 1 (defined) and * (undefined). The complementation operator COMP can be defined as follows:

$$[COMP(S)](i,j) = \begin{cases} 1, & \text{if } S(i,j) = * \\ *, & \text{if } S(i,j) = 1. \end{cases}$$

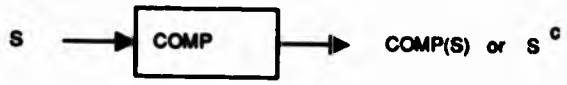


Figure 3.26: The block diagram for COMP.

The following example will illustrate the operations of EXTMAX, MIN, TRAN, NINETY, COMP. Let

$$F = \begin{bmatrix} 4 & * & 2 & * \\ 3 & 3 & 6 & 4 \\ 0 & 2 & * & 0 \end{bmatrix}_{0,2}$$

$$G = \begin{bmatrix} 2 & 5 \\ 2 & 4 \\ * & 2 \end{bmatrix}_{1,1}$$

$$S = \begin{bmatrix} 1 & * & 1 \\ * & * & * \\ 1 & * & 1 \end{bmatrix}_{0,2}$$

Then

$$EXTMAX(F,G) = \begin{bmatrix} 4 & * & 2 & * \\ 3 & 3 & 6 & 4 \\ 0 & 2 & 4 & 0 \\ * & * & 2 & * \end{bmatrix}_{0,2}$$

$$MIN(F,G) = \begin{bmatrix} 2 & 5 \\ 2 & * \end{bmatrix}_{1,1}$$

$$TRAN(G;(2,1)) = \begin{bmatrix} 2 & 5 \\ 2 & 4 \\ * & 2 \end{bmatrix}_{3,2}$$

$$NINETY(G) = \begin{bmatrix} 5 & 4 & 2 \\ 2 & 2 & * \end{bmatrix}_{-1,2}$$

$$COMP(S) = \begin{bmatrix} * & 1 & * \\ 1 & 1 & 1 \\ * & 1 & * \end{bmatrix}_{0,2}$$

3.3.3 Fundamental Digital Morphological Operation

If we assume the images are binary images which posses only two values: 1 (defined) and * (undefined), then the two image operations, EXTMAX and MIN, that are similar to the set-theoretic operations ∪ and ∩, can be represented by two notations ∨ and ∧. The operations of EXTMAX(S,T) and MIN(S,T) can be defined as $S \vee T$ and $S \wedge T$, where ∨ and ∧ are the logical symbols called cup and cap, respectively.

If S_k, $k = 1, 2, \ldots, n$, are binary images, then

$$\bigvee_{k=1}^{n} S_k$$

denotes the image that is 1 on the union of the domains of the S_k and is undefined elsewhere; similarly,

$$\bigwedge_{k=1}^{n} S_k$$

denotes the image that is 1 on the intersection of the domains of the S_k and is undefined elsewhere. Pixelwise, we have

$$[\bigvee_{k=1}^{n} S_k](i,j) = \begin{cases} 1, & \text{if there exists at least one } k' \text{ for which } S_{k'}(i,j) = 1 \\ *, & \text{if } S_k(i,j) = * \text{ for all } k. \end{cases}$$

and

$$[\bigwedge_{k=1}^{n} S_k](i,j) = \begin{cases} 1, & \text{if } S_k(i,j) = 1 \text{ for all } k \\ *, & \text{if there exists at least one } k' \text{ for which } S_{k'}(i,j) = *. \end{cases}$$

The fundamental operations are:

1. Digital Minkowski addition, or dilation

$$S \oplus E = DILATE(S,E) = \bigvee_{(i,j) \in D_s} TRAN(E;(i,j))$$

where D_s denotes the domain of S, and the domain of $S \oplus E$ equals the union of the domain of the translates, TRAN(E;(i,j)). Digital dilation results in a larger image than S wherein the small holes of S have been filled in a manner depending upon the size and the shape of the structuring element E.

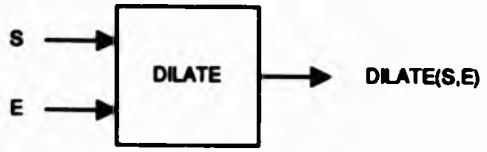


Figure 3.27: The block diagram for DILATE.

2. Digital Minkowski subtraction

$$S \ominus E = \bigwedge_{(i,j) \in DOMAIN(E)} TRAN(S;(i,j))$$

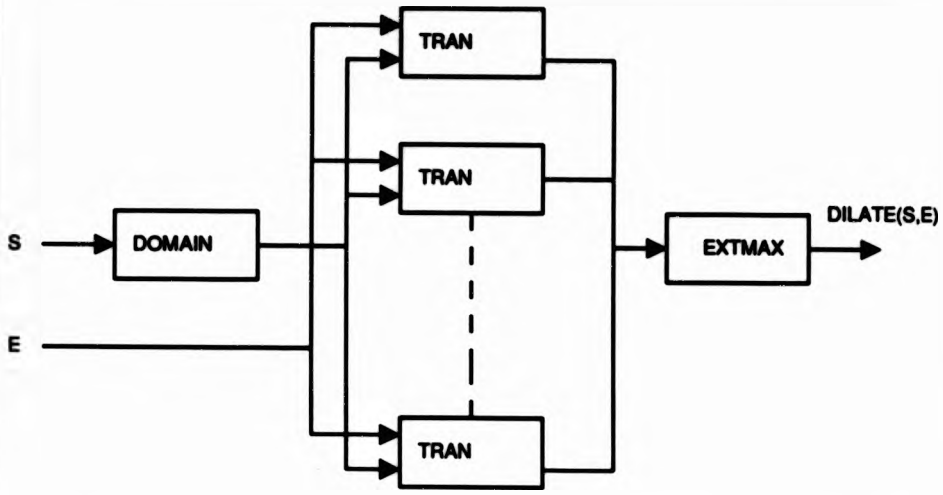


Figure 3.28: The block diagram for DILATE specified in terms of primitive operations.

3. Digital Erosion

Since $ERODE(S,E) = S \ominus (-E)$, a corresponding formulation of erosion is

$$ERODE(S,E) = \bigwedge_{(i,j)\in DOMAIN(E)} TRAN(S;(-i,-j))$$

$$= \bigwedge_{(i,j)\in DOMAIN[NINETY^2(E)]} TRAN(S;(i,j))$$

Erosion eliminates those parts of the image that are small in comparison to the structuring element. The manner of the elimination is dependent on the shape of the element.

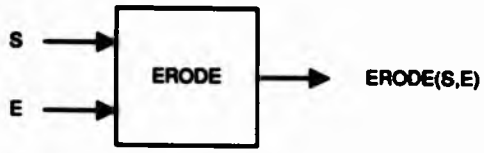


Figure 3.29: The block diagram for ERODE.

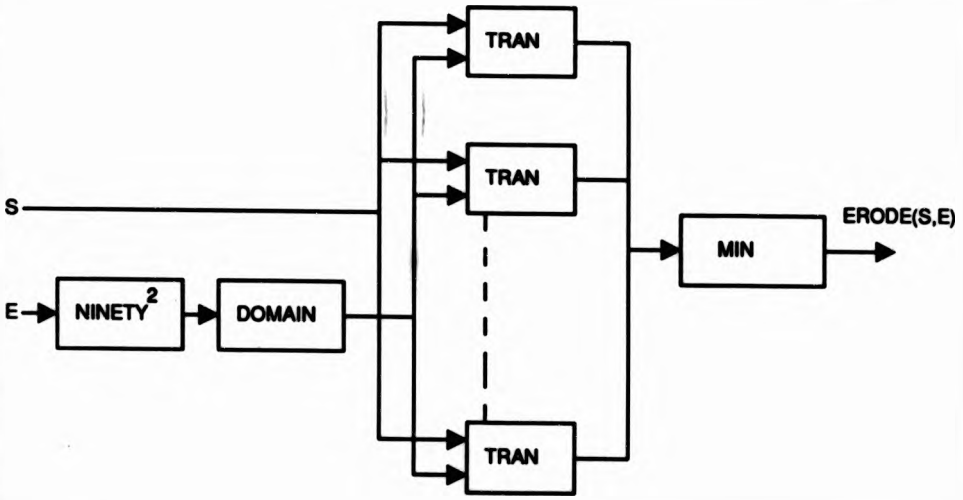


Figure 3.30: The block diagram for ERODE specified in terms of primitive operations.

4. Digital Opening and Closing

Digital opening and closing are defined analogously to the definitions of the corresponding Euclidean operators, i.e.,

$$OPEN(S,E) = [S \ominus (-E)] \oplus E = DILATE[ERODE(S,E),E]$$

and

$$CLOSE(S,E) = [S \oplus (-E)] \ominus E = ERODE[DILATE(S,-E),-E]$$

The opening can also be represented as an extended maximum (union) of fitted translates of the structuring element:

$$OPEN(S,E) = \bigvee_{(i,j)\in D_s} \{TRAN(E;(i,j)) : \; TRAN(E;(i,j)) << S\}$$

where

$$TRAN(E;(i,j)) << S$$

denotes

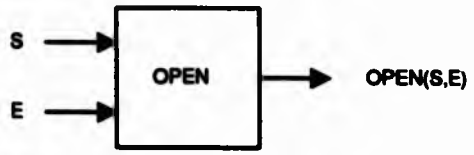
$$TRAN(E;(i,j))$$

is a subimage of S.

Opening, which can be used to determine where the given structuring element can fit as it slides around an image, has been used in biomedical image processing by [Sternberg 83].

5. Digital Boundaries

Given a binary image S, the external boundaries of S, BOUND(S), is the image whose domain is such that each pixel within it has a neighbour in the domain of S but is not in the domain of S itself. The BOUND(S) can be implemented using MIN, COMP, and DILATE as shown below.



and

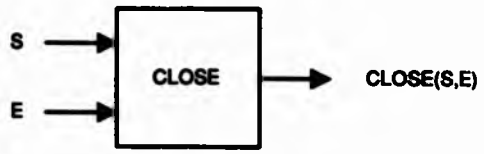
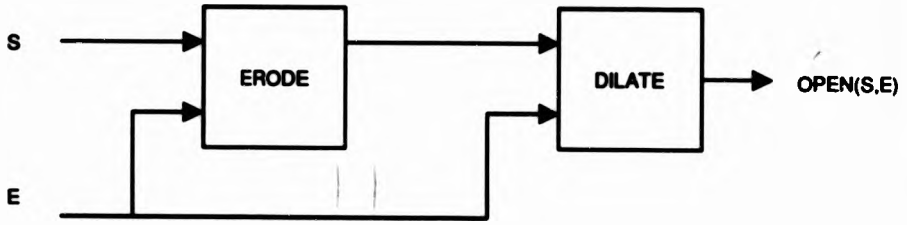


Figure 3.31: The respective block diagrams for OPEN and CLOSE.



and

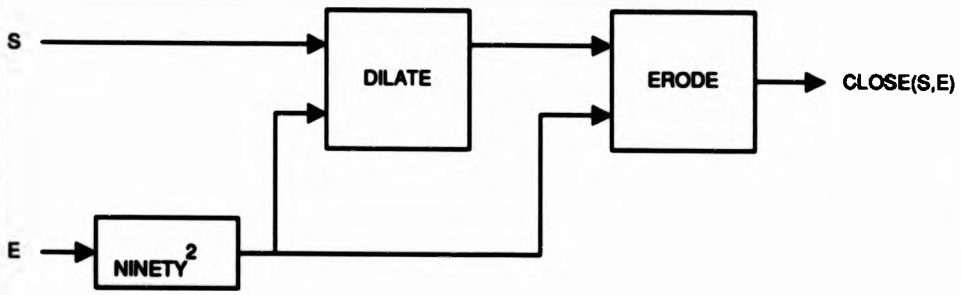


Figure 3.32: The respective block diagrams for OPEN and CLOSE specified in terms of DILATE and ERODE.

$$BOUND(S) = MIN[COMP(S), DILATE(S, E)]$$

where E is a 3×3 symmetric binary element all with the value of '1' as shown below.

$$E = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}_{-1,1}$$

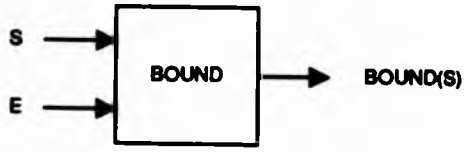


Figure 3.33: The block diagram for BOUND.

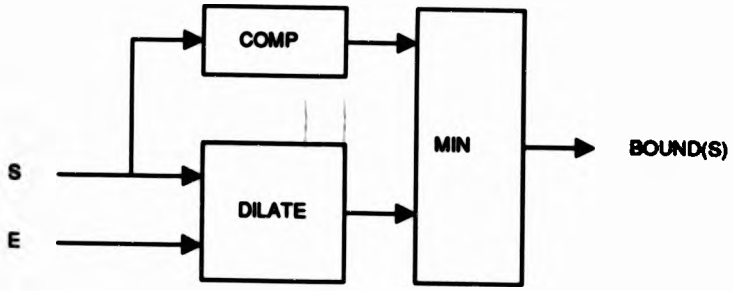


Figure 3.34: The block diagram for BOUND specified in terms of COMP, DILATE and MIN.

Parallel implementations of digital morphological operations are more efficient than sequential ones. Although we have not discussed parallel architectures, an examination of the block diagram such as dilation (figure(3.28)) and erosion (figure(3.30)) will suggest that translations simply appear as a collection of parallel operations; each is cascaded with some arithmetic operation, and then is followed by some operation acting on the parallel

outputs. Since these operations do not interfere with each other, they can easily be done in parallel. One of example in parallel approach can be found in [Sternberg 79].

3.4 Summary

The theory of mathematical morphology is rich and cannot be discussed fully in this chapter. However, this chapter provides an introduction to the basic concepts of both euclidean and digital morphological operations for image processing including the four fundamental operations of dilation, erosion, opening, and closing. Most morphological operations, no matter how complex, can be defined in terms of these basic operations. Dilation and erosion are often referred to as 'expand' and 'shrink' operations, since with dilation an object grows uniformly, while with erosion an object shrinks uniformly. They have also been used to detect edges. Opening can be used to eliminate 'salt-and-pepper' noise, while closing can be used to fill in the holes.

So far we have not discussed the effect of the nature of the structuring element used (i.e., shape and size) in the transformation. However, the nature of the structuring element itself is very important, since it will strongly influence of resulting image. For example if different sizes of structuring element are used in edge detection (figure(3.13)), different edge thicknesses will be obtained. Clearly there are a very wide range of possible shapes and sizes for the structuring element. It is impossible to predict which one is the best because the solution is problem dependent. For example, the size of the structuring element used in the noise cleaning (figure(3.11)) will depend on the size of the noise. Another example can be found in section 4.4.1, the purpose is to count the number of the leaves in the image. The first thing have to do is to break the linkages between each leaf. This can be done by appling the erosion on the image. Clearly, the number of erosions required will

depend on the size of the structuring element used: the larger the size is used, the less the number of erosions is required. Of course the size of the structuring element can not be greater than the smallest leaf size. Otherwise it can not be restored during the subsequent dilation.

Given a specific requirement, an experienced user of these techniques can choose an appropriately shaped and sized structuring element. The basic requirement is that the element selected should be kept as simple as possible to simplify the process of transformation provided that the desired purpose can be fulfilled. For example, in section 4.4.2 in stead of using a 3×3 structuring element to find the true edge, a 2×2 mask is used. Although the latter can only locate the approximate edge, it is simpler and sufficient for our purpose to illustrate the segmentation result by superimposed the edge on the original image.

We have explained the geographical properties of watersheds, catchment basin, and minima, and illustrated how these properties can be found and used as a clustering technique in two-dimensional feature space using different color bands. The next chapter will describe how an image can be segmented based on color using such techniques.

Chapter 4

IMAGE SEGMENTATION BASED ON COLOR USING THE WATERSHEDS ALGORITHM

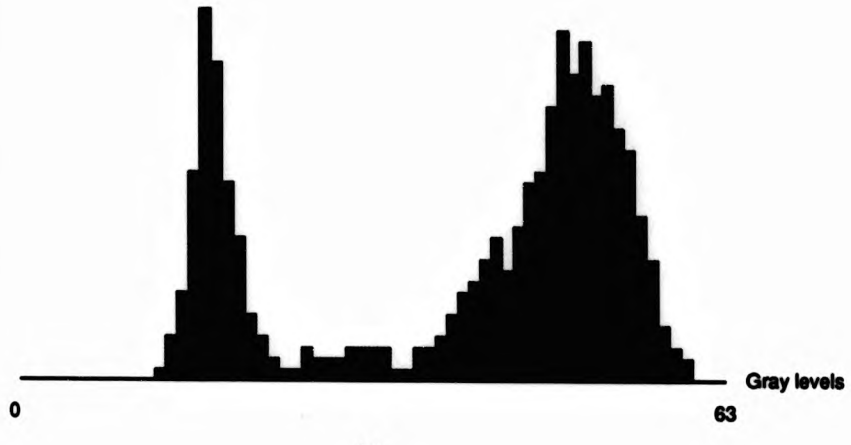
4.1 Introduction

Many techniques (discussed in chapter 2) have been used for segmentation: e.g. region extraction, edge detection, thresholding, and clustering. However, thresholding [Weszka 78] can be considered the simplest method. In segmenting a black-and-white picture, thresholding is conducted on the gray level description of the scene to produce a binary image. Usually a histogram of the gray level values of the picture elements is formed. The modes of the image histogram are identified, and pixels are labelled '0' or '1', so that edges can be located at the boundaries between connected regions of each segment.

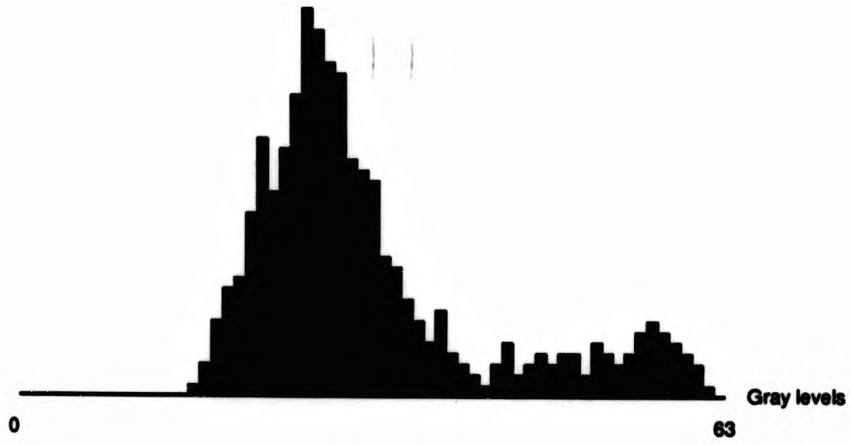
The method is to separate the peaks at the valleys, since peaks on the histogram often correspond to the gray level values of significant regions in the pictures, and then to label each pixel in the image according to which one of the peaks its intensity value belongs to. The connected pixels having the same label are obtained as the objective regions. This method can be illustrated by using the following example, a leaf on the white background (image(4.1a)), the histogram of the image (figure(4.1a)), and the segmentation result (image(4.1b)). Of course, this is the simplest case where a single thresholding is enough to separate the object and the background.

Images usually can not be segmented into more than a few segments based on a single feature of gray level alone even using multiple thresholding; if there are too many classes, they overlap and become impossible to discriminate. For example, in the natural scene (image(5.1a)), there are seven major types of regions; sky, hills, trees, lake, yacht, bushes, and shadow, but the histogram of this image has fewer than seven significant peaks (figure(4.1b)).

More refined segmentations can be obtained if we have more than a single feature (gray level) for each pixel, for example, texture and color. The reason one wants to use more features to perform image segmentation is that sometimes there are problems not resolvable with one feature that can be resolved with two or more features. This effect can be illustrated by using the two-dimensional histogram formed by Red and Green component images (figure(4.2a)), in which each concentric ellipse represents a distribution (or cluster) of image points having similar values of the color features. In this case, clusters can be easy separated. But removal of either coordinate Red or Green results in a loss of discrimination: there is a high degree of overlap of the two distributions corresponding to the two classes of regions such that no valley may exist between the modes of the



(a)



(b)

Figure 4.1: (a) Histogram of leaf, and (b) histogram of natural scene of landscape.

distributions on each of the one dimensional histogram (figure(4.2b) and (4.2c)).

However it is easy to separate the two modes in two-dimensional histogram by drawing a straight line which is a decision boundary in the two-dimensional feature space (figure(4.2a)). For many points, projection onto either the Red axis or the Green axis will give an unambiguous separation, however, there are points which can be separated by the decision surface in two-dimensional space, but not by either of Red or Green space. Thus use of the Red space and the Green space independently does not give the same power as their use together.

The feature space whose axes correspond to measurements made on an object properties must be appropriate: the different classes of regions of an image need to be represented by distinct clusters in the distribution in the two-dimensional feature space, and then the decision line can be used to separate the clusters effectively (figure(4.3a)). If the points are scattered in the space as shown in figure(4.3b), distinct clusters can not be formed. As a result there is no way to place the decision line.

When an image's properties are represented as points in this space, there are many clustering techniques which can assign them to a class by using the point's distance from an ideal point of that class, or by noting where the unknown point falls in a previously partitioned feature space. However, these methods usually involve mathematical manipulation and some prior knowledge of the image. In section 4.4 we will discuss how the watersheds algorithm (section 3.2.2) can be used to provide partitioning criteria depending only on the peaks and connectivity in the feature space (i.e. in a data-driven way without making any assumptions about the image).

How many features should be used? Obviously, there is a trade-off here: as the number of features becomes large, the discrimination improves, but the computation cost increases.

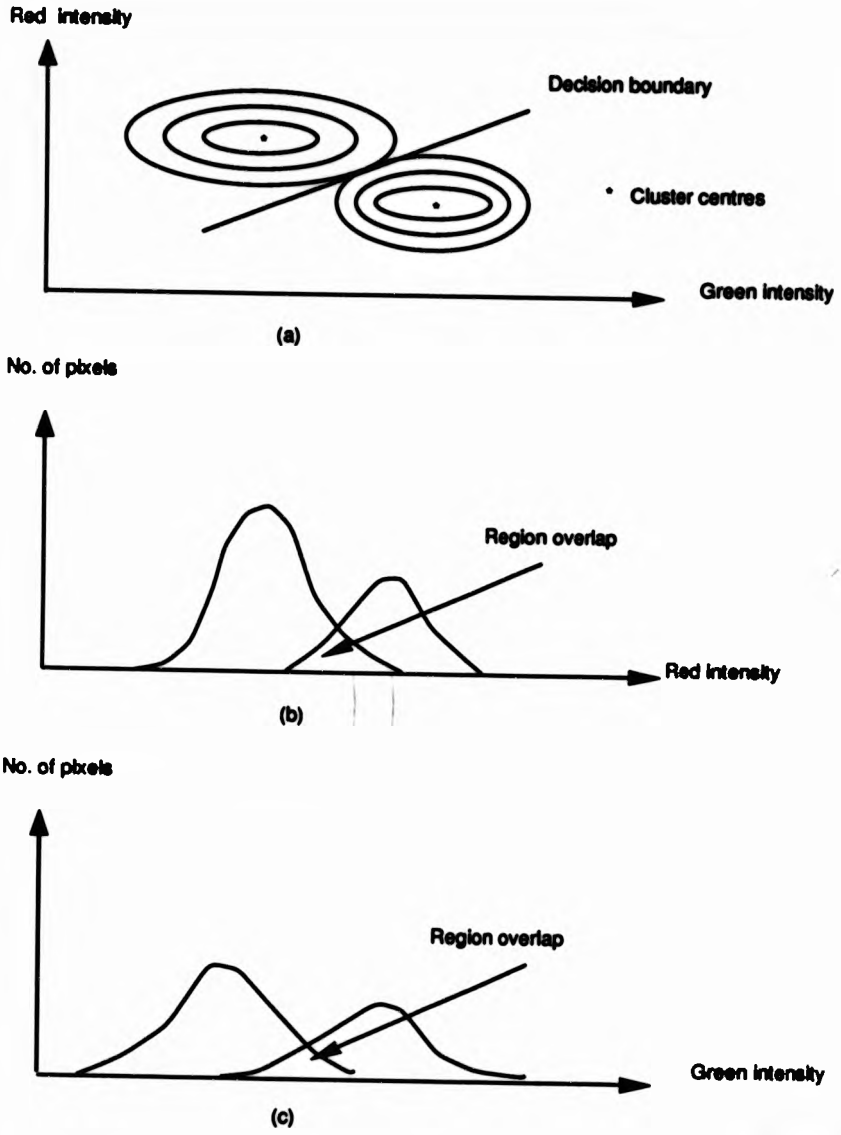


Figure 4.2: (a) Two-dimensional histogram of red and green components of a color image, (b) one-dimensional histogram of red component, and (c) one-dimensional histogram of green component.

Two other things need to be considered. Firstly, different features may extract exactly the same segments, giving no additional information for the additional computational cost. Secondly, different features may extract different and contradictory segments, and it is not easy to determine which segments should be merged, and which segments should be remained unchanged without high level knowledge such as a model of the image. So the presence of fragmentation due to multiclass segmentation cannot in general be avoided.

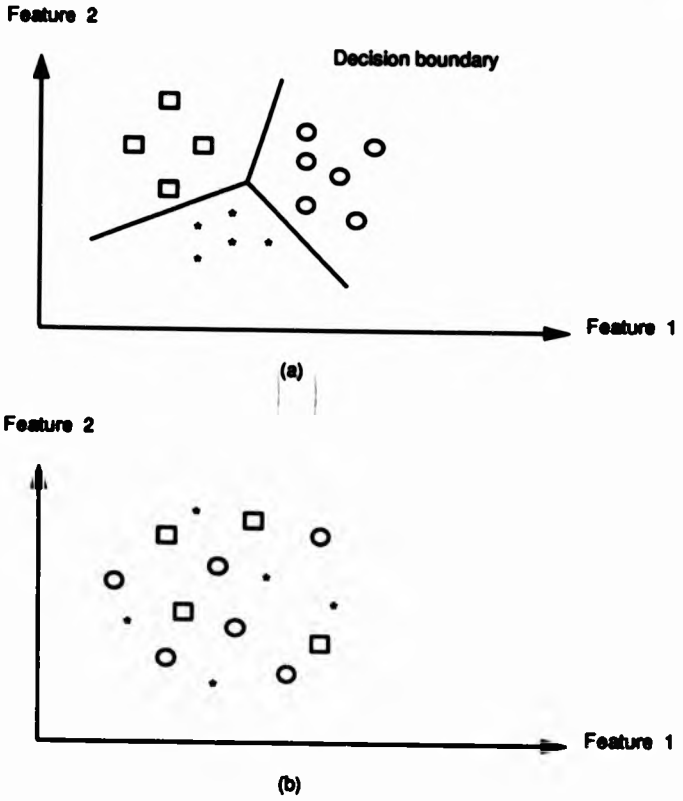


Figure 4.3: (a) Effective feature space (i.e. decision boundaries are straight lines), and (b) ineffective feature space (i.e. decision boundaries would be very complex.)

4.2 Color Description And Representation

Before starting to use color in image segmentation, let us consider what color is and how it can be represented. Light is one form of electromagnetic energy, which includes x-rays, ultraviolet, as well as infrared and radio waves. Color is a perceptual phenomenon related to human response to different wavelengths in the visible electromagnetic energy spectrum from approximately 400 to 700 nanometers (blue and red) [Ballard & Brown 82].

An object has color because of the way in which it reflects the light that is incident on it. If it reflects all wavelengths of a white light that falls on it, it will appear to be white. If it absorbs all wavelengths of a white light then it will appear to be black. If it reflects primarily the long-wavelengths of a white light, it will tend to appear to be red. However, if the light reaching the object does not contain any of the wavelength that the surface reflects, then the surface will appear black, since no light will be reflected from it at all.

The energy distribution of the light reflected from a surface is determined by the spectral response of the object and the spectral distribution of the illuminant light. The sensitivity of the detector to different wavelengths will be a third factor in determining its response. However, if the detector is the human eye, the perception of color is a psychological phenomenon rather than a purely physical phenomenon and is highly complex [Goldstein 1989][Wyszecki & Stiles 82]. Some of the important psychophysical aspects of color are briefly as follows.

The trichromatic theory states that color vision depends on three cone receptor mechanisms, each with different spectral sensitivities [Wald & Brown 65]. According to this theory, light of a particular wavelength stimulates the three mechanisms to different degrees, and the ratio of activity in the three mechanisms results in the perception of a color. This theory of color vision enables us to predict which colors should be result when we

combine lights of different colors. For example, yellow can be seen when red and green lights are mixed.

The opponent color theory states that the perception of color occurs in two principle stages [Boynton 79]. Initially, light is absorbed by (red, green and blue) cones in the retina of the eye. The neural signals originating in the retina are transmitted into a second stage of processing in which their values are summed and differenced to produce three new channels: an achromatic, or luminance, channel, a red-green opponent channel and a yellow-blue opponent channel. The three outputs from this process are transmitted to the brain, resulting in a sensation of color.

An important discovery in color perception is color constancy which states that we perceive a surface as having a constant color despite changes in the spectral composition of light reflected from it [Bruce & Green 90]. A remarkable demonstration of this issue is provided by [Land 77]. In his experiment, a picture constructed from overlapping patches of colored paper is illuminated by mixing the lights of red, green and blue. An observer perceives surprisingly little color change despite the fact that the intensity of light at each wavelength that reflect to the eye varies linearly with the incident illumination intensity at that wavelength.

Human color vision is not used in the work reported here, The intention of above discussion is to distinguish the difference between a physical and psychophysical approach to color. We are using the term color relatively loosely, and in a physical sense, to refer to the wavelength and energy distribution of reflected light.

Since a computer is used to analyse a color image, it is necessary to have some idea how color is represented in the input. A color image is typically represented in the computer as three separate arrays of numbers, with each array corresponding to the image filtered

by a (red, green, or blue) filter. The value of each element in each array represents the intensity of this component at the corresponding location. Each pixel of the color image is therefore represented by a triple of values (figure(4.4)). The (R,G,B) space can be transformed into other color spaces, such as YIQ and HSI, using linear or nonlinear transformation. Each color space has its own characteristic. For example, YIQ is an efficient method for encoding color information in TV signals; and HSI is convenient for representing human color perception.

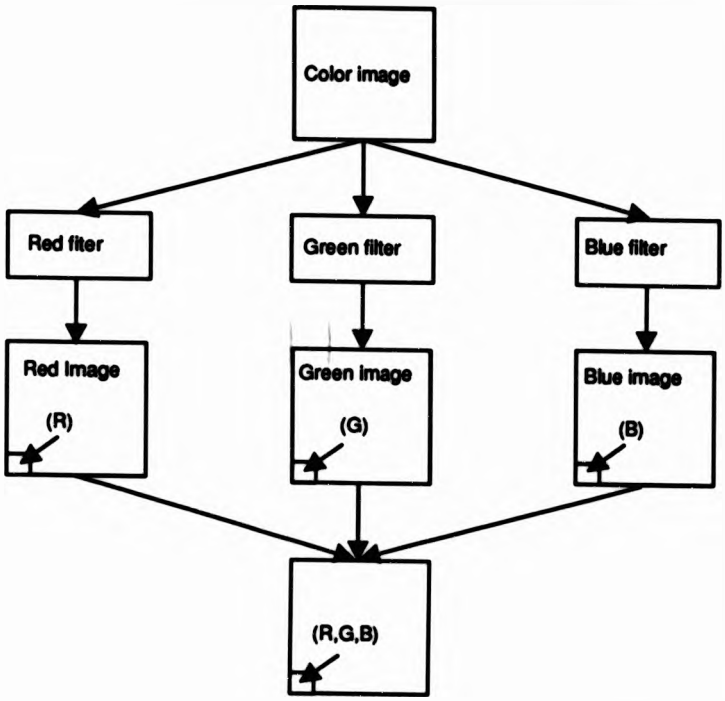


Figure 4.4: Each pixel of the color image represented by a triple of gray-level values.

The YIQ color space is used in the television system, where Y is the luminance signal, which indicates the amount of light intensity. In a black-and-white picture, the lighter

parts have more luminance than the dark areas. I and Q together are called chrominance signal. The YIQ color space is derived from the RGB color space using a simple linear transformation. The transformation equations for the National Television Systems Committee (NTSC) system [Ballard & Brown 82] is shown as:

$$Y = 0.299R + 0.587G + 0.114B$$

$$I = 0.596R - 0.275G - 0.321B$$

$$Q = 0.212R - 0.523G + 0.311B$$

This transformation is clearly invertible, so no information has been lost; it is useful for compatibility with monochrome TV, since the Y signal can be used in place of the black-and-white TV signal.

The HSI color space is based on the perceptual concepts of intensity, hue, and saturation and can be represented in a cylindrical coordinate system as shown in figure(4.5) [Kelley & Faedo 85]. Intensity is represented along the vertical axis. Angular position represents hue, and radial position represents saturation. Intensity (I) corresponds to lightness. Hue (H) is the attribute of a color perception denoted by red, yellow, green, blue, purple, and so on. Saturation (S) refers to the amount of white light in the color with the same hue. For instance pink and red are the same hue, commonly called red, but red is more saturated than pink because it contains less white light. A fully saturated color has no white light present. Normally, a color signal is divided into two parts. These are chrominance and luminance. As for the HSI color space, the term chrominance is used to indicate both hue and saturation while luminance is used to indicate lightness. The chrominance signal can be represented as a 2-dimensional vector whose magnitude represents saturation and whose angle represents hue. The luminance signal is a magnitude

representing lightness.

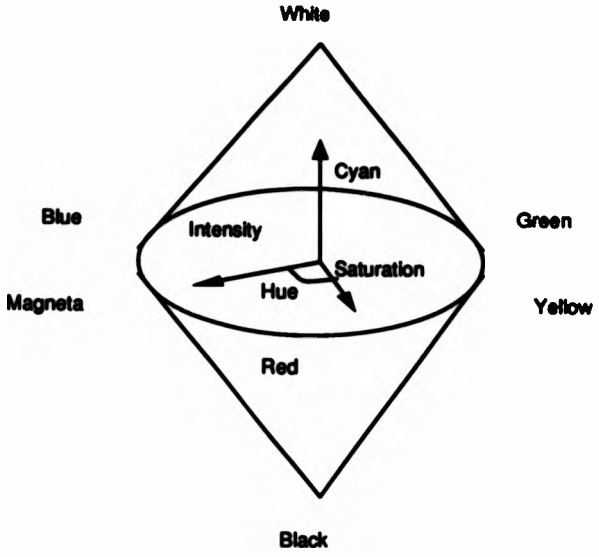


Figure 4.5: Hue-saturation-intensity color space.

In additional to YIQ and HSI, different color spaces such as YUV, XYI, and (Y,R-Y,B-Y) discussed in section 2.4 have also been used. However, the simplest color space should be the (red, green and blue) color space, since it is the output of the capturing equipment (i.e. camera) and thus does not require any transformation.

The fastest method to obtain these values simultaneously is to use a color TV-camera with red, green, and blue output channels. A multi-channel framestore is then used to digitise and store these values. Since three-channel framestore is considerably more expensive than its monochrome equivalent, a simple alternative is to use a monochrome system with an input multiplexer that can select between the red, green and blue outputs from a color camera. In this way three component color images can be captured from each

channel and stored in three different locations of the image memory successively. However, the simplest method is to add simple color discrimination to monochrome vision system by sensing the image three times with a conventional monochromatic TV-camera, each time through a different (red, green, and blue) color filter.

The main shortcoming to this method is that different tasks may require different filters. In order to provide a correct filter response for a particular task, we may need a wide variety of filters. A new approach, which can combine the advantage of an optical filter while maintaining flexibility and ease of use, is to implement the color filter characteristic electronically by means of look-up table rather than optically. This method, known as the Intelligent Color Filter [Plummer 90], can even provide a specify filter characteristics that would be physically impossible using optical methods.

In addition to the acquisition speed for the first method being roughly one third of the others, the major limitation for the latter two methods is that the image must be stationary, as the three channels are captured at different times. However, more hardware will be required for the first approach, since red, green, and blue signals need to be processed at the same time.

In our case the last method is used. A more detailed discussion about the setup for the experiment can be found in section 4.3. After capturing the image through red, green, and blue filters successively, each pixel of the image is then characterized by a set of R, G, and B color components. These components can be summarized by the following equations:

$$R = \int X(\lambda)\tau_r(\lambda)s(\lambda)d\lambda$$

$$G = \int X(\lambda)\tau_g(\lambda)s(\lambda)d\lambda$$

$$B = \int X(\lambda)\tau_b(\lambda)s(\lambda)d\lambda$$

where

- R = the intensity of red component image,
- $X(\lambda)$ = the spectral distribution of the light,
- $s(\lambda)$ = the responsivity of the camera at each wavelength,
- $\tau(\lambda)$ = the transmittance of the filter, which tells what fraction of light at each wavelength passes through the filter.

All of these integrals are evaluated over the set of wavelengths for which the filter's transmittance and camera's responsivity are nonzero.

As shown by the above equations, the information associated with each point on the color image can be viewed as a vector in 3-dimensional space [Schacter et al 76]. The (R,G,B) color space is a cube because the camera's response is bounded by zero and some maximum pixel value for each color component. In our system, each element of the vector is restricted to the range (0,63), and viewed as a vector within the 64 x 64 x 64 cube as shown in figure(4.6). The main diagonal is called the intensity or brightness axis, and corresponds roughly to the various gray levels from black to white. It is clear that the origin (0,0,0) is black and the maximum brightness (63,63,63) is white. Thus, points in the color cube get progressively brighter as one moves from the left bottom to the upper right corner. The corners of the color cube are labelled with the names of perceived colors which are formed from the three primary colors. For example, red and blue in equal amounts produce magenta when the green component is 0. Other colors are obtained when the primary colors are combined in different amounts.

Now let P denote a color vector (R,G,B) in a three-dimensional color space, which is represented by a unit cube as shown in figure(4.7). The length of the projection of P on

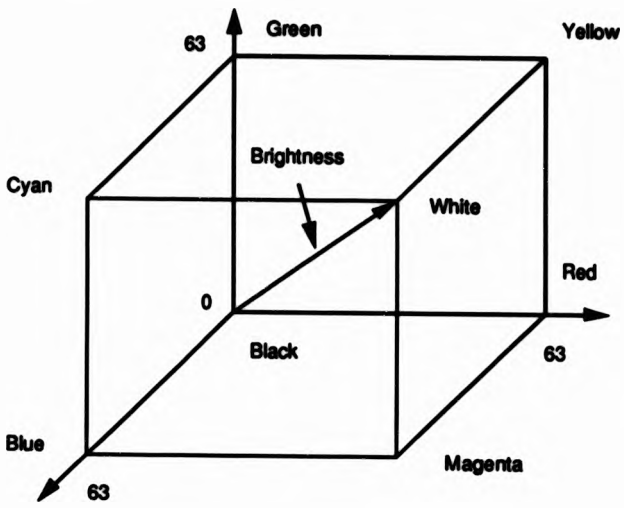


Figure 4.6: Three-dimensional (R,G,B) color space.

the main diagonal represents the brightness, and the direction represents the chromaticity (hue and saturation). The chromaticity can also be represented by the crosspoint Q of the vector P extended if required and the plane passing through the cube at the corners red, green, and blue, forming the equilateral triangle depicted in figure(4.7). The coordinate of the crosspoint (r,g,b) is called the normalized color coordinates or chromaticity coordinates given by

$$r = \frac{R}{R+G+B}$$

$$g = \frac{G}{R+G+B}$$

$$b = \frac{B}{R+G+B}$$

The points in a plane perpendicular to the brightness vector from black to white are of equal intensity. Thus the triangle shown in figure(4.7) is the largest constant intensity

plane within the cube. At any other level of intensity this triangle is smaller. The implication is that there is a smaller range of color combinations that can be formed as one approaches minimum or maximum intensity (black or white).

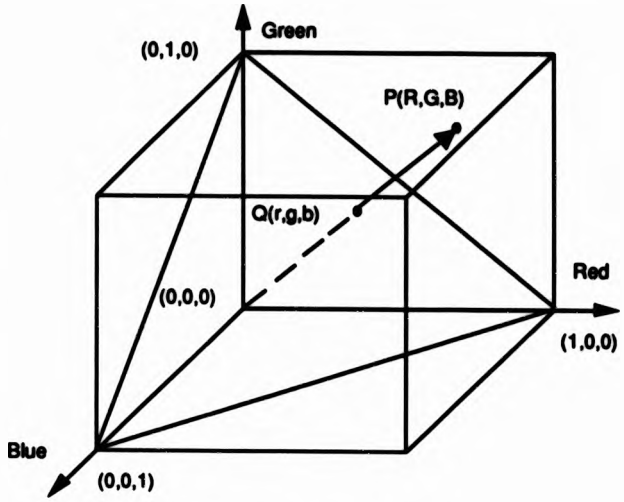


Figure 4.7: Color representation in (R,G,B) color space.

If this equilateral triangle is redrawn with unit height, the lengths of the perpendiculars from Q to the perimeter will be r, g, and b. The center of gravity of the triangle, which corresponds to $r = g = b = 1/3$, represents white as shown in figure(4.8a). This figure is a chromaticity diagram, and can now be used to describe two other characteristics of color space, hue and saturation, which are independent of intensity [Shirai 87]. The hue of a point Q in figure(4.8b) is defined as the angle θ between WR and WH, where W (white) and R (red) is the center and corner of the triangle respectively. Let Q be represented by (r,g,b); then the hue θ is formulated by using θ_1 as follows:

$$hue = \theta \tag{4.1}$$

$$\theta_1 = cos^{-1} \frac{2r - g - b}{\sqrt{6}[(r - \frac{1}{3})^2 + (g - \frac{1}{3})^2 + (b - \frac{1}{3})^2]^{1/2}}$$

where $0 < \theta_1 < \pi$

$$\theta = \begin{cases} \theta_1 & b \leq g \\ 2\pi - \theta_1 & b > g \end{cases}$$

Thus, hue can be represented as an angle θ with red as 0^0, green as 120^0, and blue as 240^0.

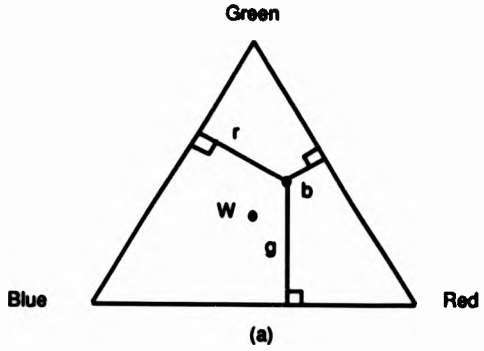
The saturation of Q is defined as a percentage of the distance of WQ to WH, where H is the crosspoint of the extension of the line WQ and the perimeter of the triangle.

The saturation S is given as follows:

$$\begin{aligned} S &= WQ/WH \\ &= 1 - 3\, min\,(r, g, b) \end{aligned} \tag{4.2}$$

The line between W and H represents the same hue with different saturation values. Thus a point close to the perimeter of the triangle represents a color with high saturation. If Q is anywhere on the perimeter of the triangle, then it has a saturation of 100% while the point W (white) is completely diluted and has a saturation of 0%.

From the viewpoint of a three-dimensional color space, each color is represented by triples of numbers representing the strengths of their red, green, and blue components. Thus the color of each pixel defines a point in the three-dimensional space. If the picture contains a large region of pixels all having approximately the same color, picture regions of near-constant color will form clusters of points within this space (figure(4.9)). If these clusters can be detected, their cluster labels can then be used to map back to the images. The points of one cluster that form connected regions in the image constitute the segments [Schacter et al 76].



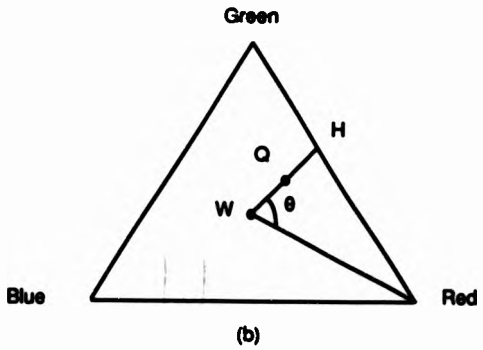


Figure 4.8: (a) Chromaticity diagram, and (b) hue and saturation.

Automatic detection of clusters in color space is much more complicated than thresholding in gray level histogram, since clusters can have complex shapes and can interact in many different ways. The process of cluster detection, can be computationally costly since it may involve searching for the optimal cluster locations and the optimal number of clusters by repeated iterations over three-dimensional space [Schacter et al 76]. One remedy for this problem is to project feature space onto a lower-dimensional space [Sarabi & Aggarwal 81] such as the R-G, R-B, or G-B faces of the color space, and look for

densely populated subregions (clusters in the face). It is reasonable to assume that such clusters will correspond to significant subpopulations of pixels. And it should be possible to segment the image effectively by choosing surfaces that separate the clusters.

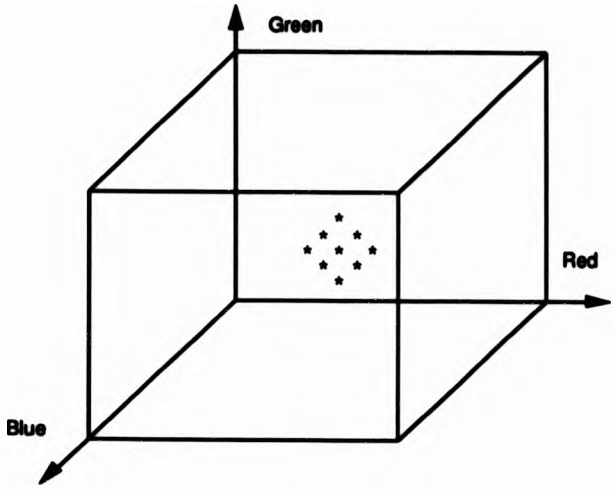


Figure 4.9: Cluster in three-dimensional color space.

The reason why three color faces are used is that sometimes there are problems in finding distinct 'modes': for example one may not be found on the R-G face, but can be found on the R-B or/and G-B face. However, if the 'mode' cannot be located on at least one of the three faces, the segment based on this mode will not be found. This effect occurs when the R,G,B colors are highly correlated. This is because the correlated colors lie in an area, very roughly ellipsoidal, near the R=G=B diagonal of the three-dimensional color space (figure(4.10)), the modes will be overlapped after projection onto the three color faces (as shown in figure(4.2)), on which the clustering algorithm will be applied.

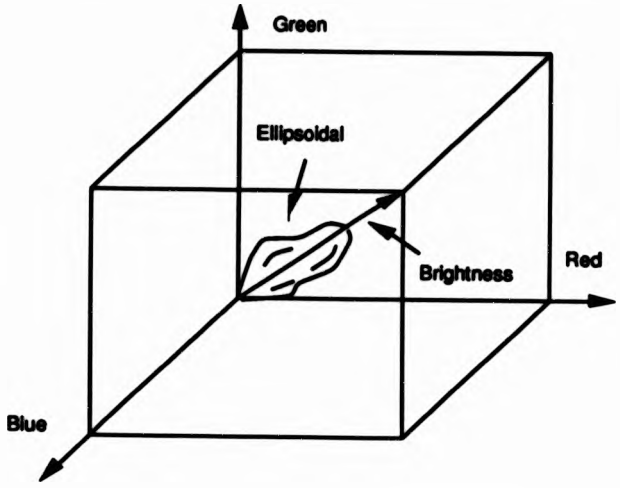


Figure 4.10: Distribution of correlated colors.

4.3 Setup for the Experiment

The setup for the experiment is shown as in figure(4.11). A JVC TK-5310 CCD black/white camera was used. The automatic gain control (AGC) in a camera is not necessarily important on a single image, but will be vital if we want to compare results over a number of images. Besides, the noise will be increased when the actual maximum output is low. Therefore, the AGC is disabled during the experiment. The color component images, which were captured through red, green and blue filters, were stored with 256 x 256 spatial resolution in the frame store. A total of twenty-four images of this size, each with 64 gray-levels, can be stored simultaneously. Tungsten filament lamps were used for illumination.

Since overlapping filters will produce the problem of correlated color as discussed in last section, a set of red, green and blue filters with non-overlapping bands are used. Their

characteristics are shown as in figure(4.12). Since a set of almost non-overlapping filters are used, the component images captured are nearly independent from each other. One can, in terms of the equations on page 76, consider the three filters and the camera to be three different detectors, with sensitivity profiles

$$\tau_r(\lambda)s(\lambda), \tau_g(\lambda)s(\lambda), \tau_b(\lambda)s(\lambda).$$

Since $\tau_r(\lambda)\tau_g(\lambda)$, $\tau_r(\lambda)\tau_b(\lambda)$ and $\tau_g(\lambda)\tau_b(\lambda)$ are all (almost) zero, the colors are only minimally correlated. Moreover, they are (largely) independent of the camera sensitivity as long as the camera's responsivity is nonzero over the set of the wavelengths for which the filter's transmittance is nonzero.

The way in which we will be using the color information (that is, use of color planes defined by rg, rb and gb pairs of detector outputs) means that the results are essentially independent of $s(\lambda)$, given that it is reasonably well-behaved. The use of broad-spectrum illumination is important, so that all of the detectors are likely to be stimulated. However, we do not need precise information on the illumination, since we are using the relative value of detector outputs in our clustering technique. In the same way, we can do without white balancing, since this would only help to label certain part of the color planes. The actual clusters would be unaffected.

The filters should be chosen so as to maximise the amount of variance in the three single-spectrum images. However, this depends on the illumination and on what is being imaged. Human color receptors overlap very considerably [Wald & Brown 65]; we require non-overlapping filters. We used non-overlapping filters with centre wavelength of 450 nm, 525 nm and 620 nm (see figure 4.12), following Shirai [Shirai 87]. This gave successful results, although the variance of the intensities transmitted through 'blue' filter was less than that of the 'red' or 'green' filters.

We will now briefly describe a few important factors when images are taken.

- The central frequencies of the filters are widely spaced in the visible spectrum. That means there will be almost no overlap of the filter characteristics. However, the choice of filters must ensure that any wavelength is transmitted through at least one filter for the technique to work effectively, since the actual wavelength sensitivity will depend on both the filters and the camera.

- The light level at the camera itself must neither be so high as to cause clipping, nor so low that random thermal noise becomes a major problem. Cameras have only a limited dynamic range to sense the brightness of the incoming light. The walls of the three-dimensional (R,G,B) color space (figure(4.6)) denote the upper and lower limits in the dynamic range of the respective color bands. If the incoming light is too bright at some pixel position, the camera cannot sense and represent it adequately and the light is clipped. The color clipping causes the saturation to the maximum pixel value (i.e. 63). If this saturation occurs in all the three color bands, the result will imply a white pixel even though the color of the light incident at the camera may not be white. A simple method to prevent this problem from occurring is to make sure the histograms of the three color component images are not saturated at the maximum gray-level value.

- The raw image often contains high frequency noise. This can be induced by the imaging system itself from errors in image digitisation or random noise produced by the camera. The presence of high frequency noise can impair the segmentation processing. We can reduce the effect of noise by taking a number of frames of the same picture and averaging the results, or applying a low pass filter to the raw image

in order to smooth out the noise. One of the problems in using low pass filtering is the selection of the smoothing window, since its size is determined both by the amount of noise that must be eliminated and the size of the smallest objects to be extracted.

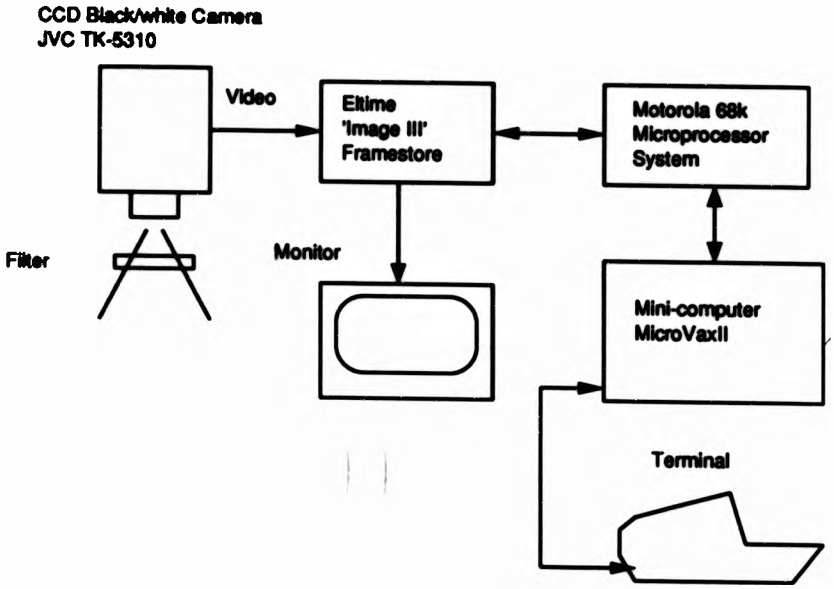


Figure 4.11: Setup for the experiments.

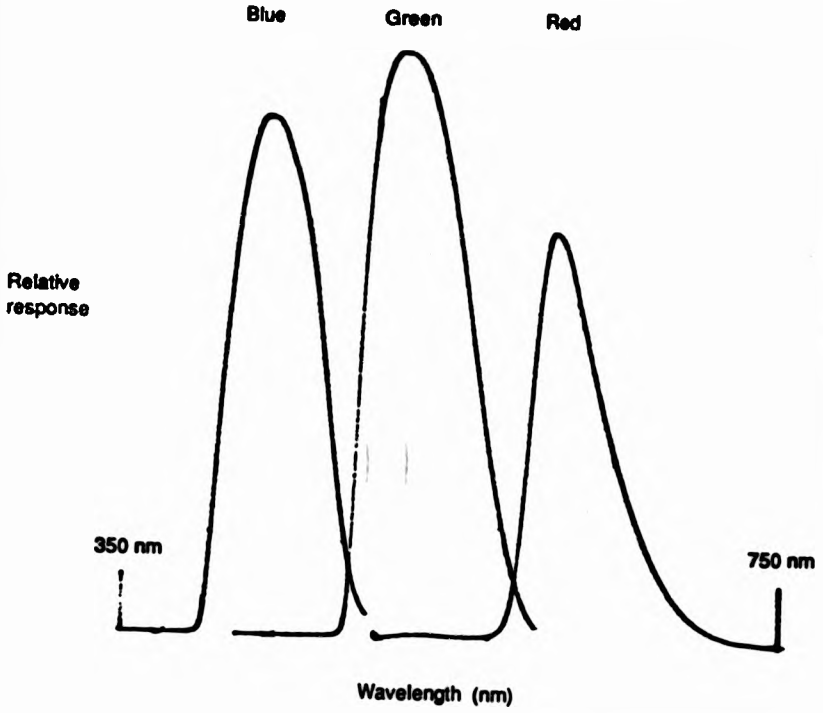


Figure 4.12: Filter characteristic.

4.4 Description of the Segmentation Method

Partitioning of a scene or an image field into disjoint regions of uniform predicate (in some sense) is often called 'Image Segmentation', as discussed in section 2.1. Most predicates used in image segmentation have relied on only gray-level information. However, color information plays an important role in human perception and recognition. Besides, color features provide additional information which can result in improvements when two regions have similar intensities, but differ in R,G,B component values. Our own experience suggests that there are many scenes in which regions are easier to identify by color than by gray level. For example, outdoor scenes which include such object as trees, sky, lakes etc. In industrial applications, there are many entities which have the same shape such as color-coded resistors, wires, and capacitors. Automatic inspection of such objects cannot be done without relating to color. Further, there are many applications where color is not directly relevant, but can be used to supply additional information to simplify and speed up the processing task.

If the predicate used depends on color, the problem is usually referred to as 'color segmentation', or 'image segmentation using color', and can be solved primarily in two ways. Uniform Color Region Extraction (UCRE) [Sarabi & Aggarwal 81][Chang et al 87][Schachter et al 76][Crisman & Thorpe 90] focuses on areas in which a color-based predicate is fulfilled, and Color Edge Extraction (CEE) [Nevatia 76,77][Robinson 76][Huntsberger & Descalzi 85] concentrates on finding boundaries.

In this chapter, the problem of color segmentation based on UCRE is approached: the algorithms proposed involve the projection of the three-dimensional (R,G,B) color space onto three two-dimensional color spaces (R,G), (R,B), and (G,B), and these are clustered using watersheds. The clusters are then used as a look-up table (LUT) to map

the corresponding images into regions of similar colors, permitting the original image to be segmented. The principal objective of these techniques that map each data point in the original three-dimensional color space into a point in an two-dimensional color space (i.e. mapping from a space of high dimensionality to one of low dimensionality) is that one can achieve a data set which can be more economically manipulated. Using the original three-dimensional color space for clustering is very computationally expensive, since searching for the optimal cluster locations and the optimal number of the clusters by repeated iterations over the data in three-dimensional space seems unavoidable. One of the examples using this method can be found in [Schacter et al 76]

Segmenting color images using clustering by watersheds in two-dimensional color space is described in figure(4.13) and illustrated in image(4.2). A color pattern which contains 128 x 128 pixels and is quantized to 6 bits in gray level shown in image(4.2a) is used to illustrate the process at each stage:

1. Red, green, and blue images (image(4.2b),(4.2c),and (4.2d)) are captured using a conventional monochromatic TV camera through three color filters.

2. Three two-dimensional histograms with 32 cluster spaces are shown in image(4.10a) to (4.10c) which are formed using:

   - red and green component images
   - red and blue component images
   - green and blue component images

   The two-dimensional histograms are smoothed to eliminate the small peaks by using a 3×3 averaging window. The average value of the pixels within the window replaces the value of the pixel being processed. The smoothed two-dimensional histograms

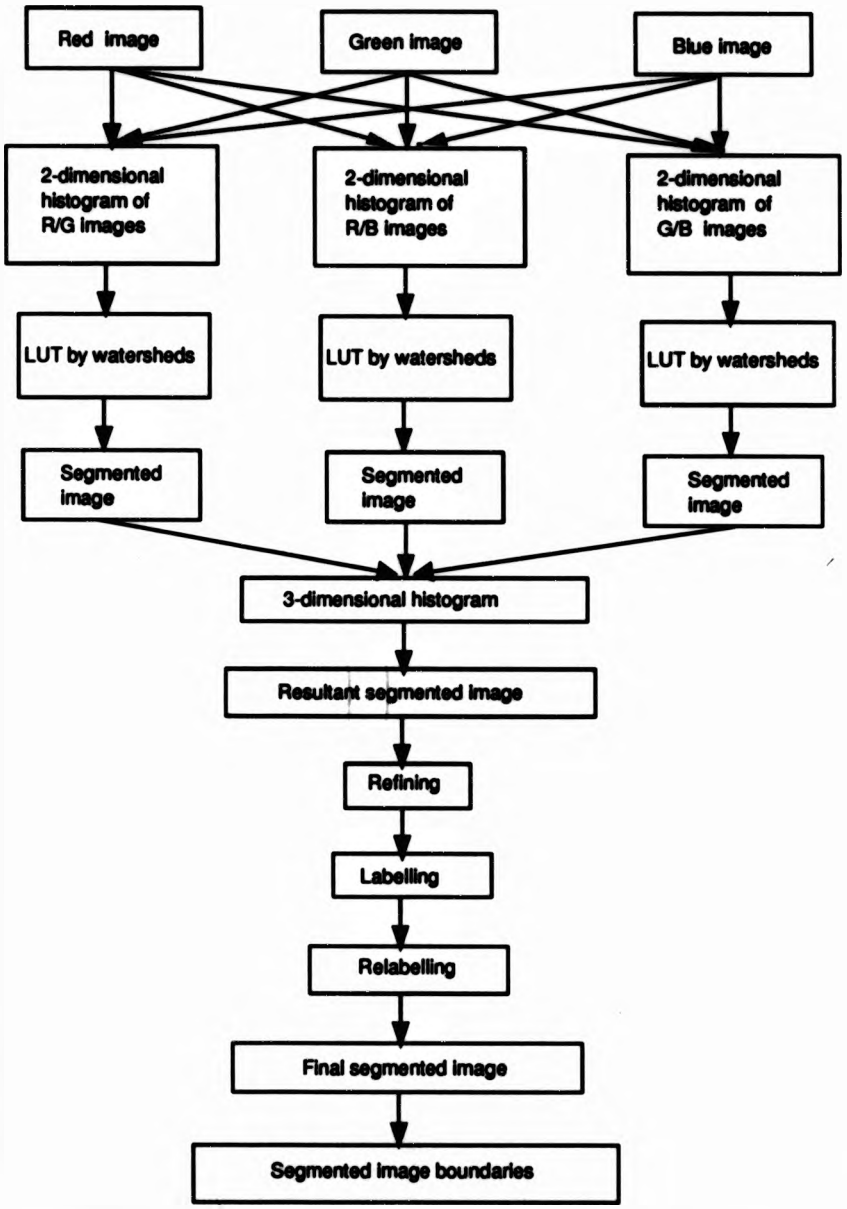


Figure 4.13: Segmentation using watersheds algorithm.

are shown in image(4.10d) to (4.10f). This smoothing is necessary because the two-dimensional histogram formed by RG, RB, and GB may be discrete and not continuous. Some locations therefore may appear as local peaks. This problem and methods of remedy are discussed in more detail in the section 4.4.3.

3. The watersheds algorithm discussed in section 3.2.2 is then used to segment each two-dimensional histogram. The peaks found in each two-dimensional histogram are shown in image(4.10g) to (4.10i). And the three LUTs created are shown in image(4.10j) to (4.10l).

4. The three segmented images based on RG, RB, and GB images (image(4.2e),(4.2f),(4.2g)), in which the segmented regions are shown as constant gray level regions, are classified by converting the individual pixel values into coordinates of the appropriate LUT so that the pixels can then be assigned to a cluster number.

5. A three-dimensional histogram can be computed from the three segmented images. Assume that the three segmented images have 2, 3, and 4 segments respectively. A cuboid of $2 \times 3 \times 4$ locations is formed. Initially, the point count at all the 24 locations is set to zero. To construct the three-dimensional histogram, the triple of segment labels of each scene point is used to index the location of the cuboid. The point count at this location is incremented by one. Note that many locations will be zero. Then the non-zero locations are labelled, starting with the largest value.

6. The resultant segmented image (image(4.2h)) can be produced by simply converting the individual label values of the three segmented images into coordinates of the cube. Each pixel can then be assigned a region label.

7. Each region label is used to generate a corresponding binary image by setting points

with the region label to '1' otherwise to '0'. Binary images are used since processing can use Boolean operators, which is faster than integer or floating point arithmetic. Besides, the binary image can be easily used for extraction of features of a region such as region shape, contours, etc.

8. The binary image is refined using 'opening' and 'closing'. Detailed discussion in these techniques can be found in section 4.4.1.

9. Extract the connected regions from each refined binary image by labelling. The refined binary image we obtain may be disconnected. As discussed in section 2.1 a segment is a connected region. The method used is based on the idea of propagation [James 87]: scanning from the top left-hand corner of the binary image, the first foreground point encountered is assigned a label. Then the scan is stopped and a region grown around this point until no connected foreground point can be found. Then the scan is resumed until an unlabelled foreground point is found whereupon a new label is used, and a region is grown. For example, the first region is labelled '1', the second region is labelled '2', and so on. This process is continued until the entire image has been scanned and all the segments labelled.

   For a binary image, one may consider two definitions of connectivity depending on the neighbour set. A '1' pixel can be defined as being connected to either its eight nearest '1' neighbours (i.e. 8-connected), or to its four horizontal and vertical '1' neighbours (i.e. 4-connected). The difference and relationship between 4-connected and 8-connected in connectivity can be illustrated in figure(4.14). For example, the region in figure(4.14a) is 8-connected but not 4-connected, but the region in figure(4.14b) is both 8-connected and 4-connected. The 4-connectedness implies 8-connectedness but not the converse. The labelled 4-connected regions of figure(4.14a)

are shown is figure(4.14c). In the case discussed here, the 8-connected version of connectivity is used.

10. Any region, whose size is less than some user-determined threshold, will be relabelled 0. We consider these small regions are either due to noise or texture. The threshold value we use is 0.1% of total image area. The pixels labelled 0 (shown by black dots in image(4.2i)) will be replaced by the label of its neighbour pixel with highest occurrence. The final segmented image is shown in image(4.2j).

11. The segmented image boundaries is shown as in image(4.2k). Detailed discussion in edge detection using dilation and erosion can be found in section 4.4.2.

1	1	1	1	1	0	0	0	1	1	1	1
0	0	1	0	1	0	0	0	0	0	1	0
1	1	0	1	1	1	1	1	2	2	0	3
1	0	0	1	1	0	0	0	2	0	0	3
(a)				(b)				(c)			

Figure 4.14: (a) 8-connected region, (b) both 4-connected and 8-connected region, and (c) labelled 4-connected regions of (a).

Image(4.2l) shows the boundaries superimposed with the black-and-white image to illustrate how the segmentation matches to the original image. The original image has eight colors: from left to right there are violet, red, orange, light green, yellow, green, blue, and the background is pink. The segmentation result in image(4.2l) is considered to be perfect. The edges shown are based on the segments found by the clustering method with no post-processing, such as thinning, linking, etc.

The result of this method is good. However, it is only suitable for images in which the color features are distinct in the color space. If not, the segments found by each two-

dimensional histogram of RG, RB, and GB may be different and even contradictory. As a result, fragmentation occurs in the resultant segmentation. This method may be useful in industrial inspection, in which we have some prior knowledge of what colors will appear in the scene e.g. color coding of resistors.

Moreover, in some situations, one or two LUTs may be enough to locate the regions in the image. For example, in image(4.2e), all the regions can be segmented except the orange area and the pink background. If we have some prior knowledge that these two colors do not require to be distinguished then only one LUT based on the RG two-dimensional histogram will do. If we look again at the image(4.2e),(4.2f) and (4.2g), we find that any two of the three LUTs can separate the eight regions. The reason why all three two-dimensional histograms are used is that sometimes there are problems where distinct modes cannot be found on RG-histogram, but can be found on RB and/or the GB-histogram. However, if the mode cannot be located on at least one of the three two-dimensional histograms, this cluster will be lost, and the corresponding segment will not be found.

4.4.1 Noise Cleaning Using Opening and Closing

The binary image resulting from mapping back the clusters on the original image often contains noise, such as isolated points, small isolated regions, and small holes in regions, which are artifacts of the clustering and do not exist in the original image. One may delete this noise by post processing the binary image. 'Opening' and 'Closing' (discussed in section 3.3.3) may be used to refine or smooth a binary image. These methods are also called geometric smoothing because the result depends on the shape of the image and the shape of the structuring element used. The refined (or smoothed) images tend to have fewer small regions.

The method consists of multiple applications of two processes: 'erosion' and 'dilation'. The purpose of the sequence of erosion is to shrink segments in a uniform manner so that small segments or small projections on segments or thin connections between segments disappear entirely. So the number of successive erosions determines the minimum size of the surviving region. The purpose of the sequence of dilations is to expand segment in a uniform manner so that small holes and concavities in segments are filled in.

Both the operations have complementary smoothing actions on the binary image, but share the shortcoming of either decreasing or increasing the size of the segment. This problem can be overcome by using the two operations together. However, the resultant smoothed image depends on the sequence of the pair of operations, which are known as 'opening' and 'closing'. Their function are summarized as follows.

1. In 'opening' (an erosion followed by a dilation) small necks between segments, and small segments and projections (presumed noise region) tend to be removed without changing the overall size of an segment: in the erosion operation an image is shrunken in a manner depending on the shape of the structuring element used. The erosion operation can be iteratively applied several times. A dilation operation can now be used to try to regrow the remaining shrunken segments to their original size. Any small noise region will have been eliminated by the erosion operation, and the larger regions will be left unaltered if the number of erosion and dilation operations are equal.

2. In 'closing' (a dilation followed by an erosion) holes and concavities tend to be filled also without changing the overall size of the segment: a sequence of dilation operation is applied first and then followed by erosion operations. Small gaps in regions and small gaps between regions (we presume the gap is caused by noise) will be filled,

but the shapes of the segment will generally be unaltered if the number of dilation and erosion operations are equal.

Image(4.3a) to (4.3c) illustrate the operation of 'opening' using the structing element of 8-connected mask. Image(4.3a) is thresholded from the segmented image(4.2h), a binary image requiring opening. The dots and the thin line beside the object region represents the noise caused by the transitional color between different color regions. After erosion, they are eliminated and the segment size is decreased uniformly as shown in Image(4.3b). After dilation, the eroded image is restored to the original size image(4.3c) with noise regions eliminated.

In large images the opening methods discussed as above can be used quite effectively. But in smaller images these methods will remove all the tiny regions, which are smaller then the structuring element used, and may break the others into many small fractions. As a result fragmentation would occur. So an alternative refining method may be used [Ohlander et al 78], where a mask of 3 x 3 pixels is used. The pixel being processed is set to '1' when more than 4 pixels in this mask is '1'. Otherwise, it is set to '0'.

Image(4.4a) to (4.4e) are another examples showing the usage of 'opening'. For example we want to count the number of leaves in the image shown in image(4.1b), which is the thresholded image of image(4.1a). The first thing we need to do is to separate the leaves from each other. This can be implemented by using opening with a structuring element of 8-connected mask as follows. After 1st and 2nd erosion the eroded image of image(4.1b) can be shown in image(4.4a) and (4.4b) respectively. After 2nd erosion, the thin connections are eliminated and the leaf size has also been decreased uniformly. In order to restore the eroded image back to the original size, the same number of erosions and dilations are required. Image(4.4c) and (4.4d) illustrate the images after the 1st and

2nd dilations respectively. After two dilations, the eroded image is restored to the original size. The number of leaves can then be found using labelling (section 4.4) and each is represented using a different gray level as shown in image(4.4e).

Image(4.5a) to (4.5c) illustrate the operation of 'closing' using the structuring element of 8-connected mask. Image(4.5a) thresholded from the segmented image(4.2h), a binary image requiring closing. After dilation, the holes in the object region are filled and the object size is increased uniformly as shown in image(4.5b). After erosion, the dilated image is restored to the original size image(4.5c) with holes eliminated.

4.4.2 Edge Detection Using Dilation and Erosion

Edges can be considered to be of three types: internal edges (being the boundaries of segment, drawn inside the segment), external edges (being the boundaries of segment, drawn outside the segment), and true edges (i.e. the actual segment boundary). This section will illustrate how these edges can be extracted using dilation and erosion (section 3.2.1). The method is illustrated in figure(4.15), where the structuring element used is a 3×3 square neighbor mask.

Image(4.6a) shows the binary image, which is part of the segmented image of image(4.2j), where black represents the area of interest. Image(4.6c) shows the dilated image. Image(4.6b) shows the eroded image. Image(4.6e) shows the external edge that is obtained by the subtraction of the dilated image from the binary image. Image(4.6d) shows the internal edge that is obtained by the subtraction of the binary image from a erosion of that image. Image(4.6f) shows the true edge that is obtained by the subtraction of the dilated image from the eroded image. This implies the addition of the external and internal edge, so the thickness of the edge is doubled. The above methods are good when

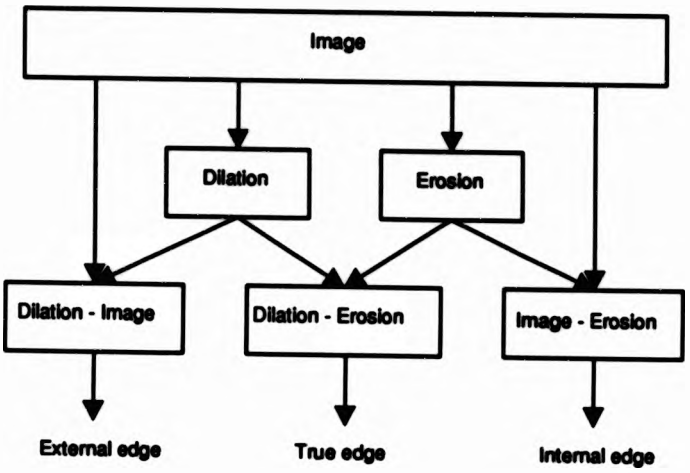


Figure 4.15: Block diagram for edge detection using dilation and erosion.

the object region and the background can be classified clearly.

For illustration, this method has also been applied to the binary image shown in Image(4.7a), which is extracted from the segmented image of image(5.3a) and is part of the lake in the landscape image (image(5.1a)). The results are described as follows: image(4.7c) shows the dilated image, image(4.7b) shows the eroded image, image(4.7e) shows the external edge, image(4.7d) shows the internal edge, and image(4.7f) shows the true edge.

If we want to find edges in a segmented image (image(4.2j)), in which each region is represented using a different label, the only edge that can usually be defined is the true edge. The reason is that the external edge of one region may become the internal edge of another region and vice versa. The above methods used to find the true edge are not very good because of their computational cost. An alternative method is to use the same 3×3 square neighbour mask used to process the segmented image. The pixel being processed is

assigned to be in a region, when this pixel and all its neighbor pixels have the same label. Otherwise it is assigned as edge. The result of using this method is shown in image(4.6g). Since the edge found is equal to the addition of external and internal edges of each region, it looks very thick.

If we simply want to illustrate the segmentation result by superimposing the edge on the original image, the following approximate method is sufficient. The pixel being processed is assigned to be inside a region, when this pixel (x, y) and its three neighbours, $(x, y+1)$, $(x+1, y)$, and $(x+1, y+1)$, as shown in figure(4.16) have the same label. Otherwise it is assigned as edge. The result of using this method is shown in image(4.2k). This method is faster than the method discussed above since the pixel being processed is compared to three neighbours only, and the edge found looks better as it is only half the thickness of the edge shown in image(4.6g). So all the edge images shown in this thesis were produced using this method.

x-1,y-1	x-1,y	x-1,y+1
x,y-1	x,y	x,y+1
x+1,y-1	x+1,y	x+y,y+1

Figure 4.16: A general 3 x 3 mask showing corresponding image pixel location.

4.4.3 Reduction of Cluster Space

The two-dimensional histograms formed by RG,RB, and GB may be fragmented and not continuous as shown in figure(4.17). For example, an image with 256 pixels by 256 pixels consists of 65536 pixels, and each of the two spectra is quantised into 64 (from

0 to 63) levels of brightness. Then the corresponding two-dimensional histogram will have 64 × 64 = 4096 locations (i.e. cluster spaces) into which counts of pixels will be accumulated. If the locations are filled nearly uniformly then a very sparse histogram may result. Some locations may appear as a local peak resulting from some isolated location being occupied by a single pixel and surrounded by empty locations; this clearly is not a true maximum, although the maximum detection algorithm, which detects locations whose pixel count is higher than elsewhere in the 3 × 3 neighbourhood, will classify it as one. As a result fragments (i.e. very small segments) will appear on the segmented image.

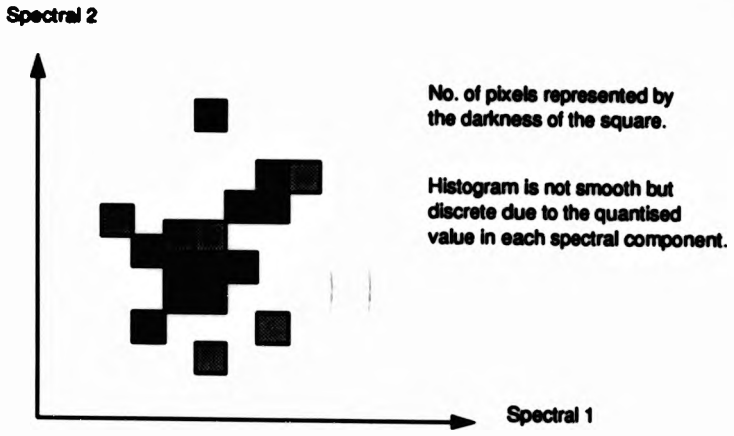


Figure 4.17: Illustration of fragmentation in a two-dimensional histogram.

Two methods can be used to remedy this problem.

1. The two-dimensional histogram is accumulated using locations which are several brightness values wide in each dimension. The dynamic range of the data in each dimension is ascertained and the cluster space is reduced.

2. The two-dimensional histogram is smoothed with a local window, in which the pixel being processed is replaced by the average value of its neighbours.

Both operations have the effect of eliminating the local peaks and tightening the clusters themselves; significant clusters remain intact while small diffuse clusters are centralized. Clearly resolution is sacrificed (i.e. colors which are only slightly different will be treated as the same), but this is necessary to yield an acceptable clustering.

However, if one classifies segmentation error into undersegmentation (i.e. failing to split regions that should be separated) and oversegmentation (i.e. splitting regions that need not be separated), then errors caused by the former will affect later processing more seriously than that by the latter, since the oversegmentation caused by the local peaks may be remedied by following a merging procedure.

It is impossible to find a general solution that specifies which cluster space or window size is the best, since the accuracy of the clustering process depends directly on how well the objects of interest on the image separate into distinct clusters as shown in figure(4.3), and it is problem dependent. That means for different kinds of images, different features may be required. In particular, color is not good for highly textured images, since the distribution may scatter widely on the two-dimensional color space and no distinct modes may exist (figure(4.3b)). However it is possible to find an appropriate cluster space or window size, if the application is limited in a particular area. Besides, fragmentation can be avoided by appending a merging algorithm at the end (i.e. the split is followed by a merging algorithm), and this is discussed in section 6.4.

The following example is used to illustrate the effect of using different sizes of smoothing window. Image(4.11a) shows the two-dimensional histogram with 42 cluster spaces formed by image(4.9e) and (4.9g). Image(4.11b) shows the peaks found in image(4.11a) without using smoothing, and the number of peaks found are 10. The segmentation is shown in image(4.12a). The segmentations using different smoothing window of 3×3, 5×5,

7 x 7, and 9 x 9 are shown in image(4.12b) to (4.12e). The effect of applying different sizes of smoothing window to the two-dimensional histogram (image(4.11a)) are shown in image(4.11c) to (4.11j).

In this case a smoothing window size of 3 x 3 seems to provide the best result, since the image can be segmented properly and the local peaks are eliminated as shown in image(4.11d) in which only 7 peaks are found. Larger values of window size result in fewer peaks. For example, if a smoothing window size of 3 x 3 is used, the isolated peaks with value greater than 8 cannot be eliminated. When the window size is 5 x 5, the isolated peaks which cannot be eliminated will be those having a value greater than 24.

In addition to the function of elimination of the local peaks, smoothing will also merge the clusters which are very close together. It is not a very serious problem when the smoothing window is small. For example when a window size of 3 x 3 is used, the clusters which will be merged together will be those having the peaks separated by a single pixel. We may consider such effect as tightening the clusters as we expected.

However, if the window size used is very large, its effect will seriously affect the segmentation since some of the well separated clusters are also forced to merge together; and the worst case is that the original distribution of the clusters may be distorted. These two effects can be illustrated in image(4.12d) and (4.12e). The former effect can be illustrated in image(4.12d). When a window size of 7 x 7 is used, there are only 3 clusters found in image(4.11h). As a result, some color chalks are no longer segmented and merge with the background. The latter effect can be illustrated in image(4.12e). When a window size of 9 x 9 is used, although 2 more clusters are found in image(4.11j) compared to image(4.11h), the segmentation is worse than image(4.12d), since the original cluster distribution is distorted.

In section 5.4.3, we will discuss how the reduced clustering space incorporated into the pyramidal data structure can be used in coarse segmentation, in which the important image regions are extracted based on the significance in the contracting two-dimensional feature (color) space.

4.5 Application in Industrial Scene Analysis

Separation of the objects of interest in the image from all other objects and the background is a key step in machine vision, itself a vital part of industrial automation. In gray scale machine vision systems, the gray level information of an image is used. Since only gray level information is used, there are a large class of image scenes which are either impossible or very difficult to analyze because of the shadow problem or the lack of attribute of color. By providing a machine vision system with color capability many of the problems that exist in gray scale work would disappear. In particular, these systems (described below) which use normalised color do not require precisely repeatable illumination.

In addition, the range of colors is usually restricted and known, and objects rarely have much variation in color over their surfaces in an industrial environment. So the color information, which is used to help isolate objects or regions, rather than to check that exactly the correct color is present, is useful for recognition, sorting or inspection of parts or products. Typical industrial application areas include:

1. Electronics; differentiating between identically shaped objects such as color coded wire, resistors, and capacitors and locating components on a circuit board [Thomas & Connolly 86][Gordillo 85][Kelley & Faedo 85][Keil 83].

2. Food production; color may also be used as a property in quality control. This is the case in food processing industry. For example, [Keller et al 86] used color and color distribution to determine quantitative measurements of 'doneness' of beef steaks by analysis the histograms of the red, green and blue color components.

3. Medicine; detecting colored pills to prevent contamination and checking color coded labels to prevent different products from mixing.

4. Mechanical; checking the assembly of colored parts and detecting faults in color paints and coatings.

Most of the above examples do not require accurate color measurement, but do need the ability to differentiate between regions of an image on the basis of color (i.e. image segmentation based on color). Therefore precision illumination is not necessary. In next section, we will discuss and illustrate how color can be used for this purpose.

However, if the color is used as a color signature to identify the region in the image having the same color, precise illumination is very important [Berry 87]. The main reason is that what we require is a parametric clustering, in which four stages are usually required and discussed as follows:

1. Decide the set of colors into which the image is to be segmented. For example, if a resistor is required to be identified based on the color coding, the set of colors is the possible colors coated on the resistor.

2. Choose representative pixels from each of the desired set of colors. These pixels are said to form a training data.

3. Use the training data to estimate the parameters of the classifier algorithm to be used. The set of parameters for a given color is called the signature of the color.

4. Using the trained classifier, classify the region of interest in the image into one of the desired colors.

If the illumination is changing during the training and classifying stage, the segmentation result will be affected.

4.5.1 Normalised Color Clustering

Let R(x,y), G(x,y) and B(x,y) be the red, green and blue components of the reflected light (raw spectral intensity) from the point (x,y) in the scene. If these values are used directly as in section 4.4, this results in segmenting the shadowed area of the scene into separate regions although both regions have the same hue. This effect can be illustrated using color building blocks, which are illuminated from the both top-left-hand and top-right-hand corners. Since solid objects are used, the shadow problem can clearly be seen as shown in image(4.8a). Spurious segmentation due to changing of intensity can also be shown in image(4.8h) to (4.8j).

The main reason is that chromatic spectra have a strong intensity component [Pratt 78], and this component varies according to the following factors:

1. the reflection characteristic of the object,
2. the distance from the illumination source,
3. the angle to the illumination source,
4. the brightness of the illumination source.

We generally want to partition the picture into regions according to the color of the surface i.e. the analysis should be unaffected by changes of illumination or the presence of shadows. For example, in industrial automation color is used to recognize, search, sort, and manipulate colored parts or color-coded objects such as wires and resistors. Segmenting a region into different regions due to shadows is not intended. An alternative method is to use normalised colors, since the chromaticity is approximately independent of the intensity. Such a segmentation will then be relatively unaffected by intensity variation, but will be

highly sensitive to hue changes. After normalising the RGB values, the intensity can be factored out [Jordan III & Bovik 88][Andreadis et al 90].

As discussed in section 4.2, the normalised RGB values can be expressed in terms of the raw spectral intensity values as follows:

$$r(x,y) = \frac{R(x,y)}{R(x,y)+G(x,y)+B(x,y)}$$

$$g(x,y) = \frac{G(x,y)}{R(x,y)+G(x,y)+B(x,y)}$$

$$b(x,y) = \frac{B(x,y)}{R(x,y)+G(x,y)+B(x,y)}$$

The ratio r(x,y), g(x,y), and b(x,y) are firstly calculated and then multipied by a constant to form the normalised red, green, and blue component image as shown in image(4.8e) to (4.8g). A constant of 63 is selected in this case. The reason for using this value is that the original maximum gray level of the image is 63. That means the maximum value of the normalised value will also be 63 when two out of three colors are zero, and the non-zero value is used as dividend. The minimum value, zero, will occur when the the color which is zero is used as dividend. However, when all the three colors are zero, the result would be meaningless. So this pixel is replaced by zero directly. After normalising the color, the normalised component images are then used for forming the two-dimensional histogram for clustering.

A region segmentation based on the normalised colors will ideally be insensitive to change of intensity. For example, suppose we have a red surface, and part of the surface is in shadow. Ideally, the segmentation based on normalised color will classify the whole surface as one region. This is illustrated in image(4.8k) to (4.8m).

The ideal shape of a histogram of a black-and-white image will have two well defined peaks. One peak on the histogram corresponds to the pixels in the background, the other

peak corresponds to the pixels in the object. Thresholding the image at the valley of the histogram between the two peaks, ensures that the correct pixels will become either part of the background or part of the object. However, when the shadow problem occurs in the image, gray scale treatment is often difficult since the image cannot be thresholded in any simple way. So the need for complex and time consuming gray scale analysis to locate the optimum threshold value cannot be avoided. This effect can be shown using the black-and-white image of the color building blocks (image(4.8a)) and the black-and-white image of the color chalks (image(4.9a)).

Besides, edge detection will also give confusing results, since some surface boundaries may be missed because the contrast across them is low, while shadows may introduce extra lines. Futher there is no easy way to distinguish the edges of objects from the edges of shadows. Moreover, the edges of the object may be blurred under the shadows and not be detected. These problems cause a lot of trouble in later processing such as recognition. For example, when the objects requiring recognition are man-made ones, simple geometric primitives such as straight lines, circles or arcs detected are used in high-level processing, in which the recognition system tries to ascertain if the combination of detected primitives is consistent with one of the known objects. Additional edges make such high-level processing very difficult.

Three edge operators, Robert's cross [Roberts 64], Sobel [Sobel 70], and Lapacian [Rosenfeld & Kak 82], are used to demonstrate this problem. Since the output of the Robert's cross operator is based on a small 2 x 2 window, it is very sensitive to noise. Sobel's operator using a 3×3 window in the computation of the gradient has the advantage of increased immunity to noise. Laplacian operator responds even more strongly to isolated point (noise point) than to edges, since its output is proportional to the difference between

the gray level of the central pixel and the average in the region. Thus in a noisy picture, the noise will produce higher Laplacian values than the edges, unless it has much lower intensity.

The edge images computed using these three operators are shown as in image(4.8n), (4.8o), and (4.8p) where Robert cross, Sobel, and Lapacian operators have been applied to the black-and-white image (image(4.8a)). None of them work very well. In addition to false edges caused by the shadows, some of the edges are blurred by the shadows and cannot be detected.

Moreover, this method can be extended to color recognition. If each region can be classified accurately on color, the color within that region can be calculated and then used for recognition. The main advantage of using this method is that the objects do not require to be placed in pre-defined positions in order to reduce the time required for searching. For example, [Kelley & Faedo 85] used N.T.S.C. Y,I,Q for color recognition. However, before these values can be calculated for recognition, the regions require to be identified. In their paper, a color resistor is used for illustration, and a rectangular search window, which matches the shape of a typical color band, is passed over the entire image in order to locate regions of saturated color. Obviously, this is a very time consuming process, particularly if the object (a resistor) has not been placed regularly. The main disadvantage of camera based color recognition is that it gives only relative results, which can be compared meaningfully only with other results measured with the same equipment.

However, transforming R,G,B to normalised components can cause instability, where small changes in R,G,B can produce large differences in the normalised values [Kender 77]. This effect can be explained in more detail as follows: the video signal is digitized to 64 levels by the analog-to-digital converter. Since there are light-colored and dark-

colored objects in the scene and all of them must be in the range of 64 levels, the value of [R(x,y)+G(x,y)+B(x,y)] is large for the light-colored objects, but small due to shadows, if we assume the factors which affect the reflected light from the objects are kept constant.

Since r(x,y), g(x,y) and b(x,y) are quotients of R(x,y), G(x,y) and B(x,y) divided by [R(x,y)+G(x,y)+B(x,y)], the value of r(x,y), g(x,y) and b(x,y) are not reliable for dark-colored objects. A small input error will cause a large output error during the transformation. For example assuming there are some error induced by the digitization, camera and represented by ε. The value R(x,y), G(x,y), and B(x,y) will becomes $R(x,y)+\varepsilon, G(x,y)+\varepsilon$, and $B(x,y)+\varepsilon$. And the corresponding corrupted normalized values will become

$$\tilde{r}(x,y) = \frac{R(x,y)+\varepsilon}{R(x,y)+G(x,y)+B(x,y)+3\varepsilon}$$

$$\tilde{g}(x,y) = \frac{G(x,y)+\varepsilon}{R(x,y)+G(x,y)+B(x,y)+3\varepsilon}$$

$$\tilde{b}(x,y) = \frac{B(x,y)+\varepsilon}{R(x,y)+G(x,y)+B(x,y)+3\varepsilon}$$

In case of $R(x,y) = G(x,y) = B(x,y) = 0$, the normalized value will become $\tilde{r}(x,y) = \tilde{g}(x,y) = \tilde{b}(x,y) = 1/3$ instead of 0.

Thus we get different normalized values. In general the values of ε is much smaller than R(x,y), G(x,y), and B(x,y) and should not have a serious effect. However, it has significant effects when R(x,y), G(x,y), and B(x,y) are not large. This can be illustrated in image(4.9k) to (4.9m). The chalk at the bottom is partitioned into two regions, since the part of the chalk under the shadow is very dark.

Two methods can be considered for solving this problem; one is to use more analog-to-digital conversion levels and other is to open the iris of the lens.

The former method involves altering the system configuration. An input video signal is digitized into 64 brightness levels and transferred to the 68k system as an image with

256×256 pixels. The resolution of imaging devices and analog-to-digital conversion greatly influences the cost and speed of the system, since the amount of data to be processed is dependent on the spatial and brightness resolutions. Ideally, higher resolution in conversion will increase the reliability and flexibility of the system, since the dynamic range over the input signal is increased and then details of the image can be represented better. For example a 6-bit converter has only 64 levels, but a 8-bit converter has 256 levels. So that if 8 bits are used in place of 6 bits, the improvement in dynamic range will be

$$(log_{10} 256 - log_{10} 64) \times 10 \; decibels \approx 6db$$

However, one thing we need to consider is the noise caused by the imaging system itself. This does not affect the output of the analog-to-digital converter seriously when the signal is much larger than the noise. However, it has a significant effect when the signal is very small. So higher conversion resolution does not bring any advantage when the noise in the system is high. Furthermore, it may decrease the speed and increase cost because of increased hardware requirements.

The latter method is easier to implement. After opening the iris of the lens, the value of dark-colored objects can be topped up. However, some of the values of light-colored objects may be boosted to such an extent as to saturate causing the color clipping effect, which can also not provide a steady result. Thus, when this method is used, proper precautions must be taken to avoid saturation occurring.

4.5.2 Experimental Results

The two test images shown in this section are color building blocks and commercial color chalks. Each component image of red, green, and blue contains 128×128 pixels and up to 64 gray levels.

Color Building Blocks

The color of the building blocks is red in top-left corner, green in bottom-left corner, blue in top-right corner, and yellow in bottom-right corner. The background is gray. The effect of spurious segmentation due to differences in intensity can be seen clearly in image(4.8h) to (4.8j). The segmentation results based on normalised colors shown in image(4.8k) to (4.8m) demonstrate how the shadow problems have been eliminated. Highlights are a problem in segmentation, since the areas of highlights do have significantly different gray level value than their surroundings. One of the methods to solve this problem is to model it and separate it from the object color [Klinker et al 87, 88, 90]. (Section 7.2.1 has a more detailed discussion about this problem.) This problem can be illustrated by the object in the top-right corner. The color of this block is blue and ideally it should be classified as one region. Due to the direct reflection of the light at the edge (image(4.8a)), part of the edge has been classified as another region.

The results on the color blocks picture are very good, since all of the three LUTs can be used to separate the five color regions. The main reason is that objects shown occupy areas of many pixels, and the colors of them are quite strong and distinct. Therefore it is easy to segment from clusters in the color space.

Color Chalks

Now let us look at a more practical problem. The test scene consists of six commercial color chalks. The colors of the chalks are green, yellow, red, brown, blue, purple, and the background is pink. The scene is illuminated from both the top-left-hand and the top-right-hand corner, so many shadows occur as shown in image(4.9a). Image(4.9h) to (4.9j) are used to illustrates the effect of spurious segmentation due to the shadows.

The segmentation results based on the normalised colors are worse than that using the color building blocks, since some of the chalks can not be classified as a single region. However, if we consider the complexity of the image, these errors should be acceptable. In the practical world, precisely correct segmentation is rarely possible, unless the image is very simple. However, the segmentation obtained is enough for many purposes. For example, we may already have prior knowledge of the scene, such as how many colors there will be, and their mean values. Then each segment can be used to calculate the mean value of color, which can be used for comparison. The results could be used for sorting the color chalks. Moreover, only one of the LUTs produced by R/G or R/B two-dimensional histogram is required to do the job, since both of them have already separated the six color chalks and the background. This can be seen in image(4.9k) and (4.9l). The LUT produced by G/B two-dimensional histogram is not sufficient to do that, since one of the color chalks is mixed with the background (image(4.9m)).

That edge detection gives confusing results can be illustrated clearly when the Robert's cross, Sobel, Lapacian operators are applied to the black-and-white image (image(4.9a)) to produce the edge images. The results are demonstrated in image(4.9n) to (4.9p), and none of them produces a satisfactory result. Since the edge images are composed of both objects and shadows, they are not easy to distinguish. It seems that adding the capacity of color to the machine vision, its power in processing of images can be enhanced in many cases by using the attribute of color, since objects which can not be separated in the conventional black-and-white image, may possibly be distinguished in the color image shown as above example. Clearly, higher level information (such as knowledge of the likely shape of the chalks) can permit precise segmentation of even a noisely black-and-white image; but here we are concerned with direct segmentation of the initial image.

4.6 Summary

This chapter describes the methods of image segmentation based on color using the watersheds algorithm to classify the image into a set of separate regions based on the predicate of color, and illustrates how the noise cleaning and edge detection can be implemented using dilation and erosion.

Using of R(x,y), G(x,y) and B(x,y) raw spectrum intensities directly for segmentation requires no transformation. But each of them is strongly influenced by intensity. Thus, spurious segmentations tend to occur because of the difference of intensity.

One of the methods which can be used to remedy this problem is to eliminate the factor of intensity by normalising the colors. Ideally, image segmentation based on normalised colors should be insensitive to change of intensity. This method is useful in machine vision, where the color is important for identification.

Precisely repeatable illumination is critical for color signature identification; however altering the illumination will move clusters, but will not, in general, change the cluster-based segmentation, so that precise repetition of illumination is not critical for this type of segmentation.

In addition to the complexity of hardware, high cost, large data storage requirement, the main limiting factor for use of normalised color in industrial application is the long processing time. Color data is three dimensional. Each of the three primary colors (i.e., red, green, and blue) is stored as an gray level image and must be processed to calculate the ratios, and that requires quite a lot of floating point calculation. The time required is often considerable, but modern CPU chips with integral floating point ALUs (e.g. 68040) make this much less of a disadvantage.

The color filters used are additive color filters which transmit one of the three primary

colors of red, green, and blue. This means irrelevant colors can be easily suppressed by using different filters [Chen & Milgram 82]. For example, blue colors are nearly invisible under a red filter. In industrial automation, this feature can be used for automatic rejection of unwanted (i.e. irrelevant to the task at hand) color objects in the background. This can also simplify the processing needed to identify the required object, and save on computer processing time.

Chapter 5

RECURSIVE CLUSTERING BY WATERSHEDS

5.1 Introduction

Recursive region splitting, which segments the image recursively can locate less noticeable regions after isolating and removing the large regions, is widely used in image segmentation [Ohlander el at 78][Ohta el at 80][Shafer & Kanade 82][Tominaga 86][Tominaga 88]. The general color features of the image used here are R,G,B color components, which constitute a three-dimensional color space. Since only a one-dimensional histogram is used, clusters which are apparent in the color space may be projected in such a manner that they are not evident on any one histogram individually. To over come this problem, additional histograms in color TV features Y,I,Q, and psychological features hue, saturation, intensity will be required [Ohlander el at 78][Shafer & Kanade 82].

The instability in transformation of R,G,B to another color space such as hue and saturation, in which small changes in R,G,B will produce large differences in the trans-

formed components, cannot be avoided [Kender 77]. As discussed in section 4.5.1 the normalized value used to calculate the hue and saturation is unstable, so the value based on these quantities is also unstable. The hue becomes unstable when the denominator of the equation 4.1 in section 4.2 tends to zero and this will occur when $r = g = b = 1/3$. That means the hue becomes unstable around the center (white) of the chromaticity diagram (figure(4.8b)). The saturation, which uses normalized values directly as shown in equation 4.2 in section 4.2, is also unstable. This will also occur around the center of the chromaticity diagram (figure(4.8b)).

As a result, the segmentation will be seriously affected, so that special precautions are required. For example, hue may not be used to compute the histogram if the saturation is near zero [Ohta el at 80]. In spite of this, the research has concentrated on using one-dimensional histograms. The reason is that clustering is much faster in one-dimensional histograms than in higher-dimensional histograms.

5.2 Description of The Recursive Clustering Method

The watersheds algorithm uses a two-dimensional histogram to detect clusters in the color space. Small clusters are often not noticed when they occur inside large regions. This is because small peaks in a two-dimensional histogram are veiled by dominant peaks, or because many small peaks overlap with each other after smoothing. Recursion is used to apply the algorithm to small areas (i.e. segments) of the image. Recursive application removes the largest peaks, allowing less noticeable peaks to become visible; the reduction in the number of peaks reduces their mutual interference. Furthermore, peaks may come from different parts of the image, and then overlap. Recursive clustering will only consider smaller portions of the image at any one time.

If the original method discussed in section 4.4 is applied to some complicated image, many fragments may be produced due to different clusters being found on different two-dimensional histograms. This problem can be illustrated using the picture of a natural landscape scene, shown in figure (5.1a) to figure (5.1h). Each component image (red, green,and blue) contains 256 x 256 pixels, and is quantized to 6 bits in gray level. In this section, we discuss how this fragmentation can be eliminated using the watersheds algorithm recursively.

In this recursive clustering algorithm, the largest region is selected. The segmentation method initially takes the whole image as a region, and using three two-dimensional histograms from RG, RB, and GB images determines the best cluster by the watersheds algorithm, splitting the region into subregions each of which is connected. The algorithm is then applied iteratively until a region is either classified as a single segment, or its size is so small that further splitting is considered meaningless. Figure(5.1) describes the procedure. The same image, the color pattern of image(4.2a), is used to explain the procedure. The process is now explained in more detail.

1. Select the largest region in an image. (Initially this region is the entire image.) If more than one region has the same largest size, any of them can be selected. In the present example, three regions in image(5.2a) have been extracted. The largest area, the background, has been reclustered and segmented into two regions as shown in image(5.2b). If no region can been found or the largest region size is smaller than 2% (a user-defined constant) of the total image area, then go to step 10.

2. Compute the three two-dimensional histograms for the portion of the image which is contained in the region.

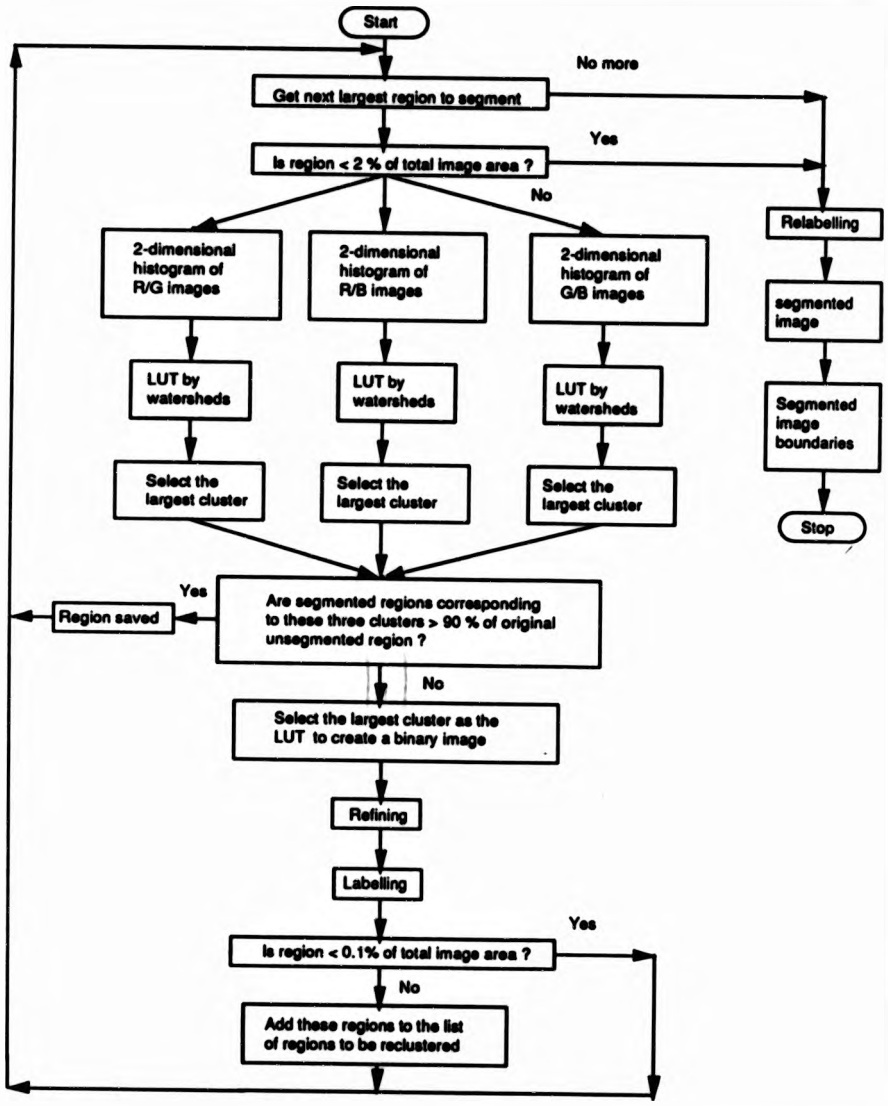


Figure 5.1: Recursive clustering using watersheds.

3. Smooth the three two-dimensional histograms using a 3×3 window as described in section 4.4.

4. Apply watersheds algorithm to these three two-dimensional histograms to creates three LUTs.

5. Select the largest cluster in each LUT. If the segmented region of the cluster is larger than 90% of the original region area, (again, a user-determined constant) then this cluster can be ignored. If all these three clusters are larger than 90% of their corresponding original region area, then this region is considered as segmented and will be saved, and we return to step 1. In the example, the first segmented region is the background, which is shown in image(5.2c).

6. Select the largest cluster from the three LUTs. Multi-threshold the portion contained in the region being segmented, using this cluster as a LUT. This generates a binary image where the points within the cluster are set to '1' and the other points are set to '0'. The reason why the largest cluster is selected is that we assume each cluster corresponds to one connected region and consider a successful segmentation to have more large regions and fewer small ones.

7. Refine the cluster. A detailed discussion about binary image refining, where small regions and thin 'necks' between large regions are eliminated and small gaps in regions are filled, using 'opening' and 'closing' can be found in section 4.4.1.

8. Extract the connected regions in the refined binary image by labelling based on the idea of propagation as described in section 4.4.

9. Save these regions for further segmentations. A user-determined size criterion, (here 0.1% of total image area) can be used to limit the minimum size of segments.

10. Continue the segmentation on the remainder of the region which is being segmented. The sequence of region extractions are shown in image(5.2d) to image(5.2j). Terminate the segmentation of the remainder regions when no more regions are larger than 2% of total image area, (again a user-determined constant).

    That means regions with less than 2% of total image area will not be used for further segmentation. Restricting the smallest size of region for recursive clustering is necessary, otherwise it may break a region into many smaller regions, which possibly represent textural elements. This is clearly not a desirable effect of the segmentation. Additionally, it is possible to limit the number of levels of recursion used, thus providing a coarse segmentation of the image.

11. Relabel the segments. Regions which are less than 0.1% of total image area (shown as the small black regions in image(5.2j), are replaced during relabelling by the label of their neighbour pixels. The segmented image is shown in image(5.2k). This is an image in which each pixel value is the number of the region containing that pixel. For example, region 10 will consist of all the pixels whose value is 10.

12. Find segment boundaries. The edge detection using dilation and erosion to find the external, internal and true edge of segmentation is discussed in section 4.4.2.

Image(5.2l) shows the segmented image boundaries and image(5.2m) shows these boundaries superimposed on the black-and-white image of the color pattern, to show how the segmentation matches to the original image. For this simple case of color pattern both methods work very well. The total eight regions can be separated accurately.

5.2.1 Experimental Results

The recursive clustering methods described in section 5.2 have been applied to a number of color images, and have produced successful results. The three test images shown in this section are a color pattern, a natural landscape scene, and a natural road scene. Each component image of red, green, and blue contains 256 x 256 pixels for the landscape scene and 128 x 128 pixels for the road scene, and all are quantized to 6 bits in gray level. The difference between these images is that the first two images are captured under the illumination of tungsten lamps on the pictures, and the third one is captured directly through the window. Segmentation results are displayed as segmented image edge representing the continuous boundaries of regions.

Color Pattern

The same color pattern (image(4.2a)) used in chapter 4 for illustration is also used here to illustrate the recursive clustering techniques. The segmentation result shown in image(5.2m) is quite acceptable, since all the eight color regions can be separated accurately without requiring any post-processing such as thinning or linking. The main shortcoming using recursive method for this case is that the processing time required is much longer than that required by the previous method due to the recursive nature.

Natural Landscape Scene

The recursive method, (in which segmented regions of size less than 0.03% of total image area will be relabeled), has also been applied on the natural scene of image(5.1a). Image(5.3a) shows the segmentation result in which the original image is segmented into 119 connected regions. The segmented image boundaries are shown in image(5.3b). Im-

age(5.3c) shows the boundaries superimposed on the black-and-white image to show how the segmentation matches the original image. Results of this nature are difficult to interpret, since no quantitative or qualitative evaluation procedure has been established for image segmentation. One possibility is the simple criterion of the percentage of pixels misclassified [Yasnoff el at 77].

Before we discuss the result of segmentation, let us look again at the image shown in image(5.1a), and find out which parts of the image that might be extracted as regions based on the feature of color. The sky, clouds, hills, and lake are relatively distinct homogeneous regions. It might be easy to segment the main area of sky, clouds, hills, and lake from the rest of the image. The difficult areas in the image are the maze of texture variations in the tree, the reflections of the hills on the lake, the shadow of the hills on the ground, and the area consisting of reflections and shadows of the yacht, which are white reflections from the yacht.

However, there is reasonable agreement between the regions obtained from the segmentation and actual regions of the natural scene. For example: the segmentations of the hills, lake, and sky are quite accurate, although none of them is classified as a single region. The main reason is that natural scenes are not as uniform as images of artificial objects (e.g. the color pattern) but rather have natural textural variations, reflectance, shadows, etc. For example, we may classify the sky as one single region, but it is impossible to do this based on the color feature alone, when there are clouds, since the color of the sky and the clouds are not the same. For this, domain dependent knowledge would be required. Besides, uniform color may not imply regions in the real world, but it provides essential elements for constructing a complete scene.

In conclusion the proposed segmenting method, which involves the recursive clustering

using watersheds, has yielded satisfactory results, if we consider the complexity of the image analysed. In fact natural scenes such as landscapes are rich in details and made up of regions that are not always homogeneous enough to be separated from one another, and that exhibit different textures.

Natural Road Scene

Another natural scene (image(5.4a)), is used for segmentation using the recursive method. The difference between this and the previous one is that the road is a real scene taken by the camera directly through the window of the laboratory. The previous one is a picture. Since the position of the picture and the lamps can be moved freely to obtain an almost even illumination and reflection of the picture, the result is almost considered as perfect. However, in the present condition the only thing we can control is opening of iris of lens of the camera. So a suitable gray level image, which is not too low to be affected by the random noise from the camera or too high to induce color clipping, can be obtained through the filters.

There were six possible regions in the scene, which are the objects of interest including tree, chair, road, roadside vegetation, double yellow lines on the roadside, and white center line of the road. The result of segmentation (image(5.4h)) might not be as good as we desired, the tree cannot be separated probably because these regions are too small to show up as a peak in the two-dimensional histogram. Additionally, the road is separated into a few regions. It can not be classified as a single region. The main reason is the problem of changing intensity as discussed in section 4.5.1. The raw red, green, and blue component image has a strong intensity component, which can be seen quite clearly when the scene displayed on the monitor. The brightness of the road does not seem to be distributed

evenly when the brightness of the monitor is being adjusted. Part of it is brighter, and the rest of it is darker. Actually, this effect appears on the whole image, the road is just more distinguished. The main cause is that the reflection from the scene is not even. Section 5.3 will illustrate how normalized color can be used to remedy this problem.

In addition to the problem of changing intensity, the texture is another problem we need to solve in order to improve the segmentation result, since the region of roadside vegetation at the bottom of the image has been divided into a lot of meaningless fragments (image(5.4h)). A possible method which can be used to solve this problem is extracting the texture parts of the image before applying the clustering. The details of this method will be discussed in section 6.3.

5.3 Recursive Clustering Based on Normalized Color

As mentioned in section 4.5.1, normalizing the color can eliminate the intensity embedded in the chromatic spectra. The simple color building blocks image (image(4.8a)), the color chalks image (image(4.9a)) and the natural road scene (image(5.4a)) are used to test the recursive methods using the normalized color. The results are discussed as below.

5.3.1 Experimental Results

Color Building Blocks

The results shown in image(5.5c) show how changing intensity affects the segmentation. The background is separated into different regions due to different values of intensity. For example, since the scene is illuminated from both top-left-hand and top-right-hand corner, the upper part of the background is separated as one region because it is brighter than the lower part. However, the lower part is also split into different regions due to the shadows

of the objects. The block on the lower-right-hand corner shows clearly how the spurious segmentation occurs. This block is divided into six different regions and the brightness of each region can be seen clearly. A total of 14 segments were found.

The segmentation results based on normalizing colors shown in image(5.5h) are quite promising. Image(5.5d) to (5.5h) show the sequence of region extracted. The four color blocks and the background have been separated accurately, which is shown in image(5.5j) using the segmented image boundaries (image(5.5i)) superimposed on the black-and-white image of the building blocks.

Color Chalks

The results of the color chalks (image(5.6c)) gives a strong impression of how the spurious segmentation problem caused by shadows. However, the results based on normalizing colors shown in image(5.6n) are very good. The only faults tend to be in areas where a very dark-colored part of an chalk is separated from it due to problem discussed in section 4.5.1. The chalk on the bottom of the image shows this effect clearly. Since the part near the upper edge of the chalk is very dark due to shadow, the problem of instability of transformation of R,G,B to normalized components exists. Thus this part of the chalk is segmented as another region. Image(5.6d) to (5.6m) are used to illustrate the sequence of extraction of separated regions, to give a clear picture how the regions are extracted one by one using the recursive methods.

Since the recursive algorithm will select the largest region to classify first, the sequence of the regions extracted also tells us the sequence in terms of region size. So the first region extracted form the image is the background shown in image(5.6d) because it is the largest. However, it is not a complete segment, since part of the background of the

central triangular region is not included. This central region was extracted and shown in image(5.6k). For display, and for further segmentation, regions are always considered to be connected. However, disconnected regions may be part of the same surface, and share the same predicate; this system will still consider them as different entities. It also explains why some of the color chalks are classified as different region although they are the same color.

This problem can be illustrated in figure(5.2) using letter 'A' and 'i'. The shaded region in figure(5.2a) would be regarded as single object with a hole representing a letter 'A' since it is connected. But the shaded regions in figure(5.2b) would be regarded as two separate objects although it is as simple as a letter 'i'. So the upper shaded region may be regarded as 'full stop' and the lower shaded region may be regarded as a numeral '1'. The reason is that they are not connected, and there is no concept of the structure forming a 'single' object (called a i). One method for solving this problem is using higher level processing where prior knowledge is incorporated such as semantic knowledge, which tells the relationship of separate things in an image, but this is beyond the scope of this thesis. Image(5.6o) shows the segmented image boundaries, and image(5.6p) shows the boundaries superimposed on the black-and-white image of the color chalks.

Natural Road Scene

This method has also been applied on the natural road scene. After normalizing the color, each ratio of r(x,y), g(x,y) and b(x,y) is multipied by a constant of 63 to form the three normalized color component image as shown in image(5.4e) to (5.4g). Then, they are used as the color feature for forming the two-dimensional histograms for clustering.

The segmentation result (image(5.4k)) clearly shows that the problem of changing

(a)

(b)

Figure 5.2: (a) Connected region representing letter 'A' and (b) two separate regions representing letter 'i'.

intensity has been solved. The road can be classified as a single region. However, the small area is sacrificed. The narrow double yellow lines at the side of the road and part of the white center line are completely missed. This is not in itself a reason to reject this segmentation method as an invalid technique, but simply demonstrates that segmentation task is problem dependent. Whether the result is good or not depends on our aim. For example, if the present task is a car which is controlled by the computer, and the camera is used to sense the outside world, then locating the region of the road for the car to run

safely on is the principle requirement. The present result might then be acceptable.

5.4 Application in Image Compression and Coarse Segmentation

As discussed in section 2.1 segmentation is useful for several problems including feature extraction [Levine 85] and image compression or data reduction [Tominaga 88]. Having discussed the segmentation methods using watersheds based on color, we will show how these methods can be used in image compression, where the number of gray levels used to represent the spatially uniform gray-level regions in the reconstructed black-and-white image is much smaller than the number of gray levels used to specify the data in the original high resolution black-and-white image. The same principle can also be applied to a color image. The only difference is the color image requires three times the memory since it is represented by red, green, and blue component images.

This section will describe how this can be done. The methods are illustrated in figure(5.3) and discussed as follows:

1. The red, green and blue images are reconstructed by firstly, calculating the mean of all the intensity levels in the same segment of the original red, green, and blue images and then replacing the value of each pixel with this mean.

2. The reconstructed color image can be obtained from the reconstructed red, green and blue component images.

3. The black-and-white image can also be reconstructed by firstly, calculating the mean of all the gray levels in the same segment of the original black-and-white image and then replacing the value of each pixel with this mean.

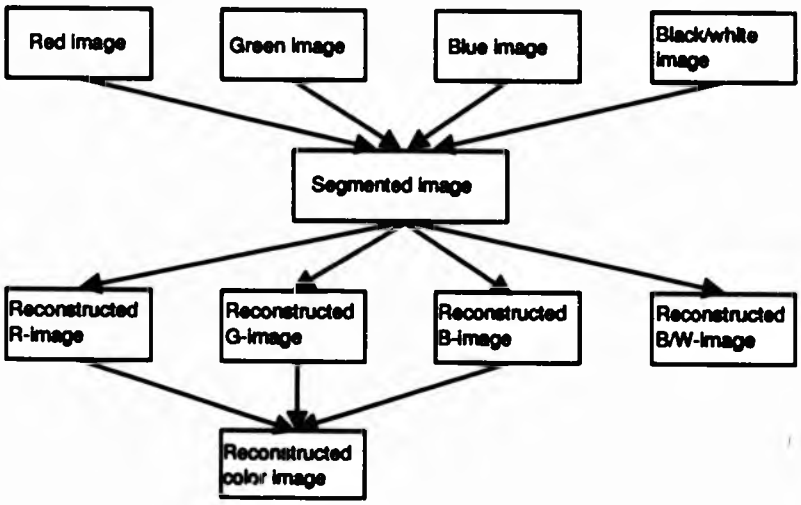


Figure 5.3: Reconstructed image using segment.

The original image is stored in a two dimensional array of size, $M \times N$, where each pixel can take K different gray-level values. So total number of bits required to store this image is

$$M \times N \times log_2 K \ \ bits$$

After reconstructing the image, there are a number of segments and each segment has the same gray-level value. That means we do not need to store the image pixel by pixel. Alternatively, it can be stored using a regional description or boundary description (section 1.4). Both methods can provide a lot of compression since the reconstructed image should be much simpler.

This effect can be illustrated using the reconstructed black-and-white image of the mandrill (image(5.7e)) and the segmented image boundaries (image(6.3e)). Comparing the result of the original image (image(5.7a)), one can see that a lot of compression has

been done as some of the regions in the reconstructed image, which are replaced by a single gray-level value, are very large indeed. Although the details such as the texture parts of the cheek and the beard are no longer available, the remaining features are sufficient for us to recognize what the image is, particularly if we have seen the original image before, or for us to learn what a mandrill looks like if we have not seen such an animal before.

We may wonder why the nose at the right hand side of the reconstructed image is not clearly distinguished from the cheek. The problem is not the reconstructed image itself, because it already exists in the original image (image(5.7a)). The reason is that although the nose is well segmented from the cheek as shown in the segmented image boundaries (image(6.3e)), the gray-level at this part is not contrasted enough and each segment of the reconstructed image is replaced by its average gray-level value of the same segment of the original image. In other words, a reconstructed image is similar to a restored image. Both of them can only be expected to be similar to the original one as much as possible, but they can never be expected to be better than the original one. The next two sections will discuss how both regional and boundary description methods are implemented.

5.4.1 Line Segment Compression

One simple regional method is as follows: since each row consists of a different sequence of segments such that the points in each segment all have the same gray-level value, the reconstructed image (image(5.7e)) can be completely determined by specifying the lengths and gray-level values of these segments in each row. For N different rows, the number of bits needed is now

$$\sum_{i=1}^{N} S_i \times (log_2 K + log_2 M) \ bits$$

where S_i is the number of segments in row i. So the rate of compression for the whole

image is

$$\frac{\sum_{i=1}^{N} S_i \times (log_2 K + log_2 M)}{M \times N \times log_2 K}$$

The compression rate for the whole image is simply the mean of the compression rate for the rows. We may thus consider the compression rate row by row. This is

$$\frac{S \times (log_2 K + log_2 M)}{M \times log_2 K}$$

Let $M = N = 256$, and $K = 64$. When all the pixels in a row is belong to one segment, i.e., $S = 1$, maximum compression can be obtained and is equal to

$$\frac{log_2 K + log_2 M}{M \times log_2 K} \approx 0.009$$

No compression can be done if the rate is equal to 1, and the corresponding number of segment is

$$S = \frac{M \times log_2 K}{log_2 K + log_2 M} \approx 109$$

The compression rate can be shown as in figure(5.4). This compression is also applicable for color image. However, this method is not suitable when the number of segment is greater than 109, since no compression is obtained. So we need to set an upper bound for the number of segments in a line. When it is greater than 109, the original method, pixel by pixel, will be used to store the image instead. That means no compression can be obtained and the line representing the compression will be clipped when S is greater than 109 (figure(5.4)). Practically, the number of segments per line is seldom greater than 109, so that this method can still provide a great deal of compression.

The rate of compression obtained will depend on how effective the coding scheme is. The method discussed above is not the optimal one, but is used only for illustration.

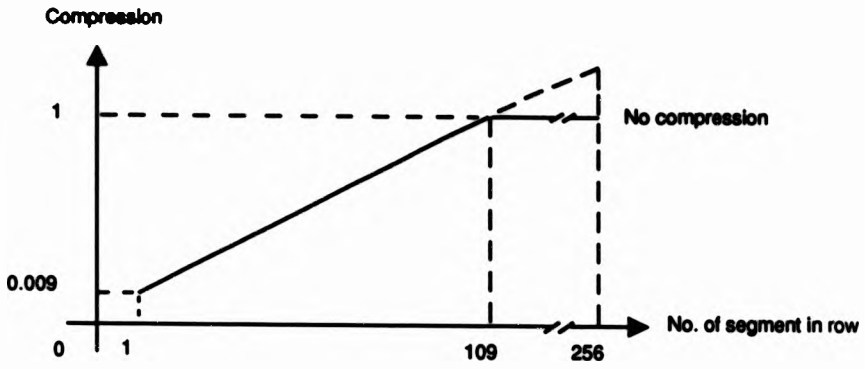


Figure 5.4: Compression using line segment.

5.4.2 Chain Code

Another possible approach is to store the description of segment boundaries using the edge image (image(6.3e)). A region is completely defined by its boundary. Thus the region can be reconstructed from it if the internal characteristic is provided (for example, gray level, color). Since the number of pixels in the whole object is considerably larger than the number of pixels in the boundary, it should be more efficient (i.e., provide image compression) to store the object by means of boundaries, than storing the object itself.

The line pattern coding technique can be implemented by using a chain code [Freeman 74]. The idea of chain code is to follow line or boundary points and to code them using a sequence, of the integers $\{0, 1, 2, 3, 4, 5, 6, 7\}$. A chain code is a string used to represent the line segment. If the size of a pixel is assumed as 1×1, each integer corresponds to a directed line segment of length 1 or $\sqrt{2}$ in the image boundary, and the directions of the segments are multiples of 45^0. Therefore the boundary of an object in an image represented in two-dimensional array is reduced to one-dimensional representation. The label of each neighbour of the pixel c shown in Figure(5.5a) gives the numeration directions

of the 8-connected chain code. Then a boundary can be specified by a string read from left to right with an arbitrary starting point and the number of the next pixel in the chain. For example, the chain code of the circle is 23456701 (figure(5.5b)).

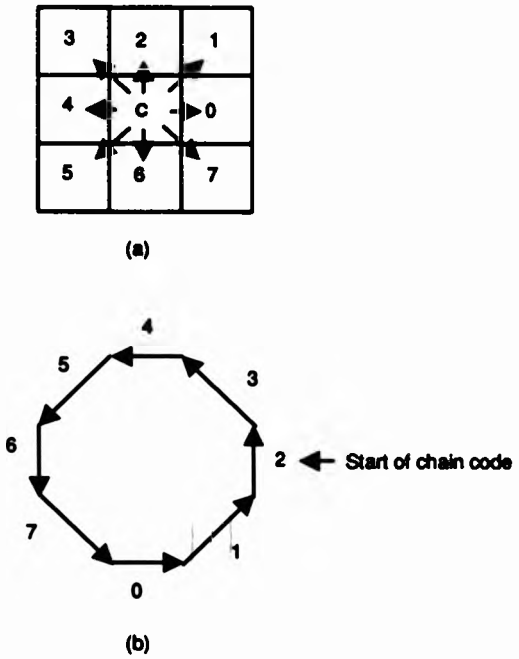


Figure 5.5: (a) 8-connected chain code, and (b) illustrated example of a circle.

Obviously, the chain code of a boundary is an efficient way of storing the description of an image, since the segmentation of an object requires only one absolute position of the object (i.e., co-ordinate for the starting point of the code) and three bits for each boundary point plus whatever is required for the label. This is considerably less storage than that required for the whole object stored pixel by pixel unless the number of boundaries is very large indeed.

In this case the number of bits needed for the background are $log_2\ K$ and for each segment are

$$(log_2\ M + log_2\ N) + log_2\ K + 3 \times E_i$$

where $(log_2\ M + log_2\ N)$ denotes the starting point, $log_2\ K$ represents the gray level (or label number), and E_i is the number of edge points in segment i.

When the number of segments are S, total bits required become

$$log_2\ K + S \times (log_2\ M + log_2\ N + log_2\ K) + 3 \times \sum_{i=1}^{S} E_i$$

So that compression rate is

$$\frac{log_2\ K + S \times (log_2\ M + log_2\ N + log_2\ K) + 3 \times \sum_{i=1}^{S} E_i}{M \times N \times log_2\ K}$$

Let $M = N = 256, K = 64$, the compression becomes

$$\frac{6 + 22 \times S + 3 \times \sum_{i=1}^{s} E_i}{256 \times 256 \times 6}$$

and really, this is $\ll 1$ even for large S and large E.

Note also that reconstruction of the compressed image is much more straightforward for the technique in section 5.4.1, and could easily be done at video rates. The technique in section 5.4.2 is (generally) more efficient, but more difficult to reconstruct - very hard to do at video rate.

If only object size and shape are of interest, further data reduction results since the internal characteristics of each segment can be discarded. So the compression rate can be simplified as follows

$$\frac{6 + 16 \times S + 3 \times \sum_{i=1}^{s} E_i}{256 \times 256 \times 6}$$

And this is a very common case in industrial usage, since the shape description of an object is frequently enough for recognizing the man-made articles.

5.4.3 Coarse Segmentation Using Pyramidal Data Structures in Clustering

Deciding which areas of an image are important is frequently a problem in low level vision, particularly if there is no higher level guidance. If the image has been transformed into a feature space, then one guide is the significant clusters in that feature space.

In this section we will show how information directly available from the pyramidal data structures in clustering can be used for extraction of such significant image structures in a solely bottom-up data-driven way without any a prior knowledge about the image. Below, we give experimental results showing how this method can be used to segment out perceptually relevant image regions. The treatment is based on the assumption that:

- Clusters, which are significant in the two-dimensional feature (color) space, correspond to relevant important regions in the image.

Pyramidal (hierarchical) data structures are a very common method used in image processing such as guided edge detection [Tanimoto & Pavlidis 75] [Levine 78]. The original image is placed on the base of the pyramid, and the ascending level is the smoothed version of the original image. For example, in guided edge detection, the low-resolution image is used to locate the edge, and the edge found on that level will be used to guide the detector to trace the edge on the next lower level. Since the low-resolution levels in the pyramid tend to blur the image, the gray-level changes that denote edges are attenuated. Thus the starting level in the pyramid must be picked carefully to ensure that the important edges are detected.

Applying the pyramidal data structure to clustering, we treat the two-dimensional histogram as an 'pseudo-image' and placed it on the base. Therefore we have a pyramid consists of 7 planes (i.e. two-dimensional histogram) stacked one upon the other, with the

original two-dimensional histogram of dimension 64 × 64 (i.e. cluster space) at the base and the 1 × 1 single cluster space at the top. The pyramidal data structure is illustrated in figure(5.6).

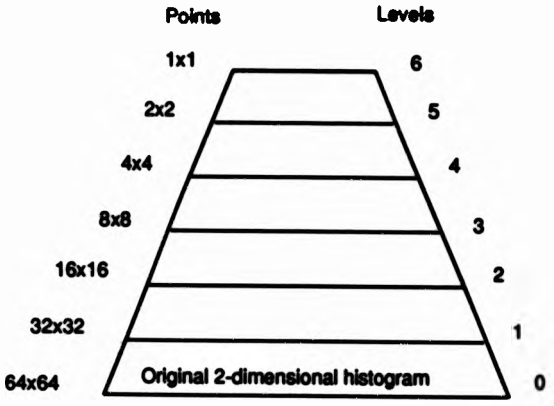


Figure 5.6: The pyramidal data structure.

The function which is used to derived an arbitrary level from next lower level is shown as follows:

Let $L_k(v)$ represent the value of a cell at level k. Then

$$L_{k+1}(v) = \sum_{i=1}^{4} L_k(v_i)$$

replacing $L_{k+1}(v)$ by the sum of its four predecessors at level k.

That means each cell at level L_{k+1} is a sum of the 2 × 2 window in the next lower level L_k. For example, each cell at level 1 is a sum of the 2 × 2 window in the level 0. But each cell at level 2 will be a sum of the 4 × 4 window in the level 0. The effect of this process is to contract the cluster space at each successive level in the pyramid.

The watersheds algorithm is originally applied on the higher level to do the clustering. Once the clusters have been found, the higher level is projected down to the original level

(i.e. 64×64). This has the effect of adding connected fringe points to the clusters identified at the higher level.

Since the planes in the pyramid are contracting versions of the previous (lower) one, clearly resolution is sacrificed (i.e. colors which are only slightly different will be treated as the same). That means fine details (in terms of color difference) tends to disappear as one ascends the pyramid. While detail may be lost, this will not matter if we are looking for the major regions of the image. There are two advantages of this method over simply applying clustering to the original two-dimensional histograms (i.e. base of pyramid) for searching the major regions of a image:

1. Computation cost savings result from the procedures not having to process the entire original two-dimensional histograms, where many local peaks may exist.

2. Local peaks due to noise are lost in the summing process going up the pyramid.

This effect can be highlighted when comparing the segmentation using the base of the pyramid to the segmentation using different levels of the pyramid. A 256×256 mandrill (image(5.7a) to (5.7d)) which has both textural regions and uniform color regions is used for illustration. Image(6.3e) is the segmented image boundaries using the original two-dimensional histograms (i.e. base of pyramid) to do clustering. The results which use three different levels (1,2,3) in the pyramid are depicted in image(5.7f) to (5.7k). Comparing the image(6.3e) to image(5.7f) to (5.7k), we see that using higher level of pyramid to do clustering, the details disappear as small segments become grouped into large entities. The reason is that the amount of texture region in the cheeks of the mandrill's face leads to a large number of local extrema. When the level ascends higher, these extrema are subsumed by some more prominent extremum. So the small clusters disappear gradually.

The segmentation of this example is good in the sense that those regions, which are given by the clusters in the reduced cluster space, really serve as landmarks of significant regions in the image. For example, the cheeks, nose, and eyes of the mandrill are well preserved (image(5.7f)). This method can be used as a coarse segmentation in which the information about the approximate location and extent of relevant significant regions in the image can be obtained. And this information can then be used in further higher-level processes such as focusing where the possible location of the desired object has been found. Thus later stage processing can be concentrated on this particular area to reduce the computational cost.

5.5 Summary

The recursive clustering techniques described in this chapter have been applied to both man made color and natural color images. Although there is no formal method for evaluating the goodness of segmentations, the results produced are quite successful since the boundaries do largely conform to the object boundaries in the image. So the quality of the segmentation is sufficient for partial segmentation, in which only the local predicate of color has been used without including any specific knowledge.

This particular technique is straightforward to justify, and not difficult to implement. But with recursive region splitting, the segmented image may consist of many small regions. So a merging algorithm may be required to combined them into a few final regions. Section 6.4 will have a more detailed discussion about this problem.

However this method provides an unbiased aid to classifying the image, since it does not require the input of any prior information, and few assumptions need to be made about the initial image. These are the user-determined constants.

The choice of the user defined constants will depend on the image itself. The 90% choice used in determining whether a region should be further segmented comes from the likelihood of discovering further peaks in the remaining 10%. Our experience suggests that these peaks will be unreliable due to noise. The limit of 2% on segment sizes is again system dependent: it depends on both the expected minimum size of segments, and on the pixel resolution of the system in use. On our system, such small regions do not cluster reliably, since they do not contain enough pixels. Similarly, relabelling of very small regions (0.03% in this case) is dependent on the likelihood of pixels actually being on color edges, again dependent on the image pixel resolution. Actual values to be used will depend on pixel resolution, and on the nature of the image itself. The precise values are not critical.

The major limitation on these methods is that the segmentation process takes a long time due to its recursive nature. Often, a task must be performed within a given time or in real time. The timing can be substantially improved by employing special hardware such as transputers which can perform high-speed parallel computation. The penalty is higher cost. However, hardware is steadily getting faster, more powerful and its cost continues to drop, so that this limitation will become less and less significant over the coming years.

An illustration of how segmentation may be used for image compression was presented. Methods based on both regions and boundaries have been considered. The region method used to code the image is very straightforward. Instead of storing the original image pixel by pixel, we store the length of the segment in each row and the gray-level value of that segment. A maximum compression rate, 0.009, can be obtained, when all the row belongs to one segment. However, nothing can be gained if the number of segment is equal 109 in a row; and more bits will be required when the number of segment is greater than 109.

The boundary method used is 8-connected chain code, which provides the shape description of the area of interest. If the absolute starting point of the chain code and the internal characteristics of the segment are provided, the image could be reconstructed. However, the shortcoming of both methods is that compression can not be obtained as anticipated if fragmentation happens on the segmentation. For example, in the worst case 65536 segments occur on an 256 x 256 pixel image. The best way may be to store the image pixel by pixel again without doing any compression.

Finally there is a discussion showing how a pyramidal data structure can be used in clustering for coarse segmentation, in which the important image regions based on the significance in the two-dimensional feature (color) spaces are extracted. The mandrill image is used to illustrate this method and the results are quite promising, since the important regions such as the cheek, nose, and eyes are extracted successfully.

Chapter 6

IMPROVING IMAGE SEGMENTATION

6.1 Introduction

We have discussed how the homogeneity of color is used as the criterion (i.e., predicate) for dividing the image into regions. If we assume that a meaningful segmentation, where each region corresponds to a part of an object surface in the input image, is a successful segmentation, it is hardly ever achieved by using only such local image properties. The reason is that coherent regions are not always equal to meaningful regions in the practical world, unless the image is very simple and the local properties can be related to the interesting objects. For example, a brown wire in an electrical plug represents live. In this case, successful segmentation on that color means finding that wire. Otherwise higher level processes, where various additional information about the image must be employed, is required to obtain useful regions. However, this chapter will discuss some possible modifications which can improve the results before they are passed to further processing.

Some modifications, which include pre-processing and post-processing, can be used to improve the segmentation. So far, the segmentation has restricted itself to the use of color information, pixel by pixel, alone. Here, we consider how additional information, edge and texture, can be used to improve the segmentation. The edge points and the textured parts (busy parts) in an image, which are apt to be divided into a lot of meaningless fragments, can firstly be extracted from the image during pre-processing. Then the segmentation process is applied only to the parts remaining. After the segmentation is completed, the edge points are replaced with the label of one of its neighbours during relabelling. However, the textured parts are considered as terminal regions. A merge algorithm can be used for post-processing to further reduce the fragments by merging them with a neighbouring region.

6.2 Edge Points Extraction

Edge points, which tend to contain strange, transitional colors due to spatial averaging, are likely to affect the clustering [Connah & Fishbourne 81]. This effect is most likely to happen on edges with high gradient magnitudes [Broder & Rosenfeld 81]. The gradient magnitudes tend to be higher on different colored objects and backgrounds than in the interiors, so that high magnitudes are associated with colors that are often intermediate between the colored objects and the background. Suppose the color image is used to form a two-dimensional histogram consisting of objects of one color on the background of another color, the color pixels with high-gradient magnitudes should produce clusters lying between the object and the background clusters. So by suppressing those pixels whose color gradient magnitudes are high from the two-dimensional histogram, we should improve the separation between the clusters, since suppressed pixels are likely to have

colors intermediate between those of the object and the background. These intermediate colors are partly due to resolution effects: the actual color is not a color found in entity being imaged itself, since the pixel corresponds to part of both object and background.

How the transitional color affects the segmentation is illustrated in image(4.2i). Most of the regions requiring relabelling are edges between two different color regions. Since the edge regions usually are discrete and small in size, they can be almost eliminated during relabelling. However, one could simplify the clustering process by computing the edge points, that is the sharp changes in intensity or color, and then extracting them from the image before segmentation.

There are two methods of incorporating edge information into a clustering algorithm: find the edges in each of the color component image, and determine an edge in the color image if certain relations between edges in individual components are satisfied, or find edges in the intensity image. Although it is quite simple to compute the edge in a color image [Robinson 76], this approach seems computationally inefficient since the gradient is computed effectively three times.

Moreover, the edges obtained from the chromatic spectra are rarely very different from those obtained from the intensity image: this is not surprising, as the chromatic spectra have a strong intensity factor and the color edges are highly correlated with the edges in the corresponding intensity image [Pratt 78]. Thus, the second approach has been chosen since it will require less computation and the edges obtained are quite satisfactory for this application.

When edge detection is used as a segmentation method, an edge detector is firstly applied to an image to obtain a gradient image. Then a threshold is applied to the gradient image to obtain an edge image. An edge point is said to be present at a pixel

if its result exceeds this threshold. In practice, this set of edge points will often have gaps due to blur or noise, so that they will not characterizes a boundary completely. Thus edge-detection algorithms are typically followed by edge linking procedures, such as Hough transform, relaxation method, designed to assemble edge points into a meaningful set of object boundaries. So in such a situation, a local threshold or a set of thresholds is generally used to obtain the edge image. For example, the threshold used by [Haralick & Dinstein 75] is the mean of the gradient values of a local region. One of the biggest problems in using edge detectors is that human intervention is usually required to select these threshold values when one would prefer the whole process to be automatic.

However, in the present situation, it is not intended to produce a closed boundaries that will completely surround the regions. The objects of interest are only the sharp edge points that lie on the boundary between two high contrast regions. This is why the Robert's cross operator and a global threshold are used.

The Robert's cross operator is based on a 2×2 window, and detects a distinct change in intensity between two adjacent points in the gray level image. So only very sharp edges with high contrast between the surfaces which form the edges will be detected. Edges which are formed by a gradual change in intensity across the edge cannot be detected. However, the result is quite sensitive to noise and surface irregularities, since a small window is used for the computation.

Using the Robert's cross operator, we pre-process the intensity image to find a gradient image. After thresholding the gradient image, a binary image can be obtained, representing the edge image. If a pixel is found to be an edge point, it is then ignored in the clustering process. The procedure consists of:

1. Applying the Robert's cross operator to a black-and-white image to produce a gra-

dient image Grad I(x,y).

2. Thresholding the gradient image to produce an edge image E(x,y) as follows:

$$E(x,y) = \begin{cases} 1, & \text{iff } T < Grad\ I(x,y) \\ 0, & \text{otherwise.} \end{cases}$$

A global threshold T used by [Ohta 85] is chosen. The cutoff value T for the gradient image is determined from the mode value and the standard deviation of the histogram for the gradient image.

$$T = mode\ value + standard\ deviation \times 1.4$$

The amount of edge visible in the edge image E(x,y) is directly controlled by the choice of T. Larger values of T result in fewer edge with values near T=0 yielding many edges. When the value of T is set to 0, all the edges on the gradient image will also appear on the edge image.

3. Edge points found in the thresholded gradient image are then ignored in the clustering process. After completing the segmentation, these points will be relabelled.

This method detects areas with sharp edges. It is not sensitive to areas where the edges are formed by a gradual change in intensity at boundaries between objects. This method has been applied to the color pattern (image(4.2a)). Image(6.1a) and (6.1b) show the sequences of the extraction of edge points. Image(6.1a) is the gradient image computed using Robert's cross operator on image(4.2a). Image(6.1b) is the edge image. Image(6.1c) is the final segmented image. Image(6.1d) is the segmented image boundaries and image(6.1e) is the segmented image boundaries superimposed on image(4.2a) to show how the segmentation matches the original image.

6.3 Textured Parts Extraction

Texture contains important information about the structural arrangement of surfaces and their relationship to the surrounding environment. This information is useful in several applications. Changes in texture can be used to segment an image [Tomita & Tsuji 90]. For example texture measures can be used to classify agricultural crops in remote sensing.

The quantitative definition of texture does not exist. However, a texture may be defined as a repetitive arrangement of a basic pattern, which yields an characterization of textures as smooth, rough, fine, coarse, and so on. The repetition and the basic pattern can be entirely regular or there can be an element of randomness in each or both. A texture normally has some basic pattern and some degree of randomness in repetition although there are textures which show no obvious fixed basic pattern nor repetition. So an effective texture finding operator must have certain properties. It should describe the spatial distribution of the texture elements (randomness), and characterize the shape and the size of the texture elements.

The three principal approaches used in texture analysis are statistical, structural, and spectral [Gonzalez & Wintz 87].

The statistical approach usually describes texture by statistical rules governing the distribution and relation of gray level [James 87]. For example, first order statistics are based on properties of single pixels (i.e., the means, variances, and the standard deviations). An example of a first order statistic in that it depends on individual pixel gray level is the gray level histogram. Second order statistics describe how properties vary in pairs of pixels, third order statistics consider triplets and so on. Since most statistical methods apply statistics to gray level value of pixels, this approach is more suitable for images that do not have a geometrical regularity such as microtextures.

The structural approach attempts to break the texture down into its basic pattern and describes such patterns in terms of rules for generating them [Ballard & Brown 82]. Formally, these rules are grammars of different types, such as tree grammars, shape grammars, and web grammars. Thus these basic patterns and the grammar can be used for such purposes as classification or comparison of textured images. This approach is best for describing textures, which have been imaged at high resolution, and where there is much regularity in the replication of the basic pattern. Sometimes there is no clear distinction between these two approaches, particularly when the organisation of the basic pattern is essentially random. In such a case, we may need to describe this randomness by some statistical properties. Thus we have returned to the statistical case. A good survey on statistical and structural approaches to texture analysis is given in [Haralick 79].

Spectral techniques are based on properties of the Fourier spectrum [Bajcsy 73]. It is based on the result that any real periodic function of the original texture has a symmetric Fourier spectrum with respect to the original. If a function is periodic, then its Fourier spectrum could provide a representation of the image, and the corresponding features, for example: high-energy, narrow peaks .., derived from the Fourier spectrum, form a good description of the periodic pattern. Further, an interesting property of the Fourier spectrum is that it is invariant with respect to translation of pattern in the spatial domain, but not with respect to rotation. Thus the directionality of a pattern in the picture is preserved in the Fourier spectrum. So it is possible to distinguish directional and non-directional components of texture.

Segmentation of textured parts (busy parts) in an image is difficult using only color features [Ohta 85], since they are apt to be divided into many tiny, meaningless regions (image(5.4j)). This problem can be solved by eliminating textured parts from examination

by the segmentation process in a similar way to the removal of edges, discussed in section 6.2. Pre-processing is required to extract the textured parts from the image and the segmentation process is applied only to the parts without texture.

The decision on what type of texture analysis to choose depends very much on the specific application. However, one of the simplest textural operator is an edge detector, which is not intended to produce the exact boundaries between regions, but merely indicate areas of rapidly changing intensity. The 'busyness' of an image can be measured by counting the number of edges per unit area [Rosenfeld & Troy 70][Rosenfeld & Thurston 71]. This texture measure can distinguish coarse and fine textures; coarseness is related inversely to the amount of edge per unit area: coarse textures have a small number of edges per unit area, and fine textures have a high number of edges per unit area. This can be used to extract regions of similar texture.

The method proposed by [Rosenfeld & Troy 70][Rosenfeld & Thurston 71] is applying an edge detector to the image to produce a gradient image. Then an average operator which produces values proportional to the number of edges in a local region is applied to it. An image whose intensity is proportional to the edges per unit area of the gradient image is obtained. The image in which each pixel's value is edge per unit area is actually a defocused gradient image. Finally, the textured parts can be obtained by thresholding the image to separate busy areas from quiet ones. However, the result of this method is not good for the present purpose, since average values of sharp edges may be high enough to cause them to be thresholded as busy areas.

[Ohta 85] used a similar measure to aid him in pre-extracting the textured area from the image before segmentation commenced. The modified methods are: first applying an Laplacian edge detector to the green image to produce an gradient image, second

thresholding the gradient image to produce an edge image on which '1' represents the edge found. Then utilizing a 9 x 9 window on the edge image, if more than a pre-defined number (say 5) of 9 subwindows (3 x 3) have at least one '1', the central pixel is considered to be textured. He noted that this method can detect an area with scattered edge segments and is not sensitive to the sharp edges at boundaries between objects because sharp high contrast edges are spatially localised and this technique will ignore them.

6.3.1 Description of the Pre-processing Texture Extraction Method

The method used measures a texture feature along with the R,G,B color features. When each segmentation is commenced, the texture feature will be the first feature used for clustering. If the largest cluster found could fulfil the specified requirement (the segmented region of the cluster is larger than 90% of the original region area), this region is considered as highly textured area and such area would be classified as terminal and be saved. Otherwise, the region would then be processed using the color features. This approach provided a reasonably smooth integration of texture and color information within the recursive region-splitting algorithm. The texture feature chosen, based on an image's gray level and local busyness, is that used by [Ekstrom et al 84]. The method illustrated in figure(6.1) is performed by the following steps:

1. Apply the Robert's cross operator to the black-and white image to produce a gradient image Grad I(x,y).

2. Use the black-and-white image and the gradient image to compute a two-dimensional histogram for the portion of the image that required segmentation.

3. Smooth the two-dimensional histogram to eliminate their local peaks by using a 3 x 3 average window. The average value of the pixels within the window replaces

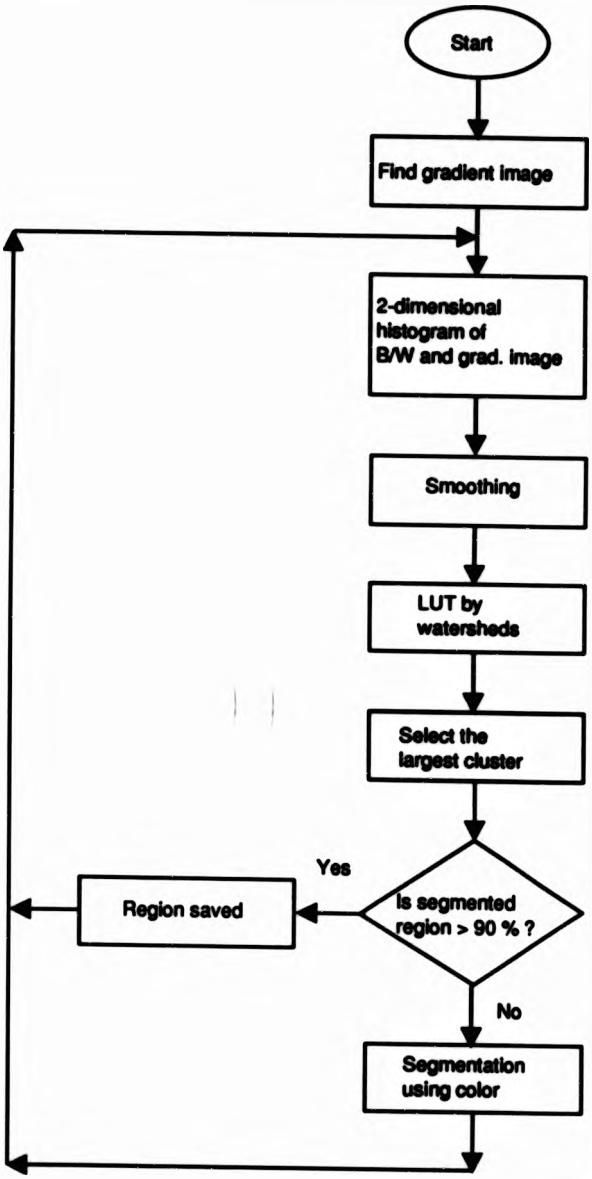


Figure 6.1: Texture extraction based on gray level and local busyness.

the value of the pixel being processed.

4. Apply watersheds algorithm to this two-dimensional histogram to create a LUT.

5. Select the largest cluster in the LUT. If the segmented region of the cluster is larger than 90% of the original region area, this region is considered a terminal region and will be saved. Then go back to step 2.

6. Segment the region using color features as described in section 5.2.

This method has been applied to the natural road scene (image(5.4a)), and the results are shown in image(6.2a) to (6.2d). Image(6.2a) is the gradient image computed using Robert's cross operator on image(5.4a). Image(6.2b) is the segmented image. Image(6.2c) is the segmented image boundaries and image(6.2d) is the segmented image boundaries superimposed on image(5.4a) to show how the segmentation matches the original image. Comparing image(6.2d) with image(5.4j), the present results are very good, since the vegetation at the bottom of the image which produces a lot of texture can be nearly classified as a single region. But in image(5.4j) many fragments occur. However, the effectiveness of the method can not be guaranteed. A more detailed discussion about this problem and the remedied method can be found in next section.

Since an additional texture feature is used in the segmentation, and each region will be examined using this feature before color segmentation is commenced, the computational cost is increased. A slightly modification can be used to reduce the computational cost. The method is to segment the image with color feature as in the original technique. If a choppy and noisy segmentation result occurs, the texture feature is used instead [Shafer & kanade 82]. In this method, texture features can be used to override color features only where necessary. However it is indeed very difficult to define a choppy and noisy condition.

6.3.2 Description of the Post-processing Texture Extraction Method

This section will discuss post processing for texture extraction, a method used to extract the textured regions which can not be extracted using the method mentioned in the previous section. Let us consider the following example: image(6.3f) is the segmented image boundaries superimposed on the original black-and-white image(5.7a) using color incorporated texture extraction applied to image (5.7a) to (5.7d). However the fragmentation, which is caused by the textured parts on the cheek of the mandrill, still exists in the image. It can not be extracted as a single segment. The reason is that the texture extraction discussed as above does not work effectively since no significant clusters occurs in the brightness and local busyness feature space.

The unsuccessful pre-texture extraction will create not only fragmentation but also a subsequent problem that will upset our relabelling method, because, during the relabelling processing, the unclassified pixels will be labelled by the neighbour with most frequent occurrence. This method works very well when the regions requiring relabelling are small and are scattered throughout in the whole image. However, the regions shown at the top-left corner and right hand side of the nose in the image(6.3a), requiring relabelling are very large. (The transitional edge effect can also been seen in the contour of the nose.) There are no simple ways to decide which neighbour label should be used without incorporating external knowledge. So we may prefer to classify such regions as a single one and pass it to higher level processing rather than merging it with its neighbour at this stage.

One might wonder why these regions can not be extracted during the segmentation, since their sizes are much larger than the threshold value. The problem is that during each clustering, after opening and coloring, the small regions which are less than a user determined threshold will be assigned to '0', and will be replaced by their neighbour label

during relabelling. However, when the clustering is applied on a highly textured area, fragmentation results in a lot of very small isolated areas. These areas may eventually join together to form such a large region.

A possible method for solving this problem is to use a smaller threshold value. Then the regions requiring relabelling will be reduced. As a result no such a large regions requiring relabelling will exist. However, fragmentation will occur in the final segmented image because a lot of small regions are allowed to exist. This is unlikely to be suitable for further processing. Further, there is no simple way to decide on the threshold value since the clustering is data-driven, no prior information about the image having been assumed.

Another method which can solve this problem is post processing for texture extraction, which is achieved as follows: after completing the color-based segmentation, apply the following steps to the binary image segments which require relabelling. (The example used to illustrate this method is the mandrill image where 218 segments have already been found.)

1. Opening the binary image (image(6.3a)) in which pixels requiring relabelling have been set to '1'.

2. Regions larger than the threshold value will be treated as a segment (image(6.3b)) and will be added to the segmented image to form the final segmented image. In other words the large segments found can be considered as highly textured parts in the image (i.e. regions at the top-left hand and right-hand side of the nose). In this case the threshold value is 20 pixels and 31 regions are extracted.

3. Regions of size less than a user determined threshold is set to '0'.

4. The pixels set to '0' are replaced by their neighbour label during relabelling.

Under these circumstances, the small region can be relabelled to reduce the fragmentation in the final segmented image, and highly textured parts which can not be extracted during pre-processing can be identified here as shown in image(6.3b). Image(6.3d) is the segmented image boundaries and the total number of segments found is 249.

However, the problem of what threshold size should be the optimal value chosen still remains unsolved, since too small a size implies too many segments, and too large a size implies details lost. How we balance the pros and cons depends upon the application, on what we want from the segmentation and on the higher level processing intended.

This effect can be shown in the image(6.3b) and (6.3d). The right eyeball can not be extracted since it is less than the threshold value (i.e. 20). If the threshold is decreased to 18 pixels, the eyeball can be extracted as shown in image(6.3c) in which 39 pixels are extracted (i.e. eight more regions extracted). And the total number of segments shown on image(6.3e) become 257. From this example we can conclude that the smallest size allowed to exist will depend on the smallest object of interests and this will certainly change from image to image. Even using the same image, different sizes may be required for different purposes. Additionally, if the threshold value is correctly chosen, regions of interest will already be extracted during the recursive clustering process and will not appear on the image(6.3a). The question is how we can know this size before starting the segmentation.

How false edges are produced by textured areas is illustrated in image(6.3g), (6.3h) and (6.3i) which show the computed edges using Robert cross, Sobel and Lapacian operators respectively. All of them produce many busy edges. Although there is no standard measurement to assess the results, the boundaries shown in image(6.3e) seems to be superior intuitively. Besides, these images also show how the region-based segmentation method works better than the edge-based method, since the latter needs further processing such

as edge-linking, or relaxation to find closed boundaries.

6.4 Recursive Merging

If we consider a segmentation successful on the grounds of having more large regions and fewer very small ones, one method which can be used to improve the result is a split followed by a merge algorithm. Fragmentation (that is, a image is divided into a large number of small regions) may occur due to the process of recursive region splitting, although we have limited the smallest size of the region used for reclustering. For example say there is a region whose size is greater than the smallest limit, say 2% of total image area. After reclustering, it may be segmented into a lot of small regions with size smaller than 2% but larger than the size which we considered as noise. However, this problem can be ameliorated by appending a merging algorithm at the end, i.e. using region splitting followed by merging algorithm. A merging procedure, such as the one based merely on color difference between two adjacent regions, will remove most of the fragmentation, and improve the performance of the segmentation. Problems can arise in the selecting the merging criterion.

In the region merging process, adjacent similar regions are merged until no two adjacent regions are sufficiently similar to be merged. One of the principal criteria of the region merging algorithm is the similarity of regions: the difference of the average levels of the adjacent regions. If the difference is less than a threshold, regions are merged. Other criteria sometimes involves object-dependent heuristics, when man-made objects are present in the scenes [Brice & Fennema 70]. If objects in scenes are not constrained, an object-dependent heuristic cannot be used. Parvin [Parvin 84] used a merging criterion where two adjacent regions were merged if they had equal mean values. The test of equal

mean value was achieved by analysis of variance from a subset of pixels in each region.

It is clear that there is no easy way to determine any single threshold for region merging that is acceptable, even for a simple scene. Thus, dynamic setting of thresholds is required in the different regions, but that is a complex process to automate without global guidance or prior knowledge. However, if the purpose is to perform an initial segmentation without use of external knowledge, and what we want is to have more large regions and fewer small regions, a very simple criterion discussed below can be used. After segmenting the image, we define the smallest size of region which should be outlined. Any region which is less than that size will be eliminated and merged with its neighbour region with smallest color difference to obtain the final segmented image.

Let the chromaticity be represented by color $C(r,g,b)$ in the three-dimensional R,G,B color space. The difference of the two colors $C_1(r_1,g_1,b_1)$ and $C_2(r_2,g_2,b_2)$ can be defined in different ways, three of which are as follows [Ito & Fukushima 76]:

1. $D_1(C_1,C_2) = \{(r_1-r_2)^2+(g_1-g_2)^2+(b_1-b_2)^2\}^{1/2}$

2. $D_2(C_1,C_2) = |r_1-r_2|+|g_1-g_2|+|b_1-b_2|$

3. $D_3(C_1,C_2) = max\{|r_1-r_2|,|g_1-g_2|,|b_1-b_2|\}$

Two colors C_1 and C_2 are defined to be equivalent if and only if the difference between two vectors is less than a certain threshold C_t. That means C_t determines the discrimination ability for the color. Method (1) takes more computation time, but can discriminate different colors more precisely. Method (2) and method (3) can only yield approximate solution. In method (3) two colors C_1 and C_2 are defined to be equivalent if all of $|r_1-r_2|$,$|g_1-g_2|$ and $|b_1-b_2|$ are less than C_t, and not to be equivalent if any of them is equal to or greater than C_t. The decision of which difference measure should be

used depends heavily on the problem. In my case, method (2) was adopted because it is sufficient for the current purpose and its computation overhead is small.

The reason why method (3) is not used is that if we consider the cluster is the points formed within a certain range in the three-dimensional color spaces shown in figure(4.9), the solution of method (3) expressed in terms of the largest vector difference is not precise enough because only one direction is shown. However, the solution of method (2) expressed in terms of three directions can give us a general idea that the difference between points and a cluster in a three-directional spaces.

The merging methods illustrated in the block diagram (figure(6.2)) can be described as follows: after segmentation is completed,

1. Go through the segmented image looking at the smallest region which is less than some pre-defined region size. If no such region can be found, merging is complete.
2. Calculate the average color of red, green and blue of that region and its neighbours.
3. Calculate the color difference between that region and its neighbours using method (2).
4. Merge that region to its neighbour with the smallest color difference. Go back to step 1.

Image(6.4a) to (6.4c) and image(6.4d) to (6.4f) show the results of recursive merging the segmented image of a landscape scene shown in image(5.3a) with region size less than 0.06% and 0.09% of the total image area respectively. Image(6.4a) and (6.4d) are the new segmented images. Image(6.4b) and (6.4e) are the new segmented image boundaries. Image(6.4c) and (6.4f) are the new segmented image boundaries superimposed on image(5.1a).

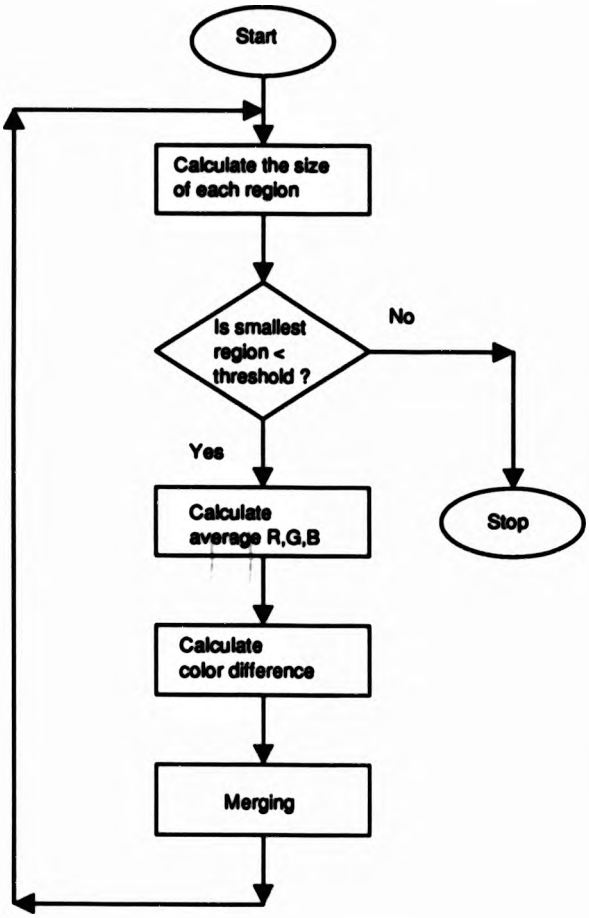


Figure 6.2: Recursive merging based on region size and color difference.

To assess the effectiveness of the merging algorithm, then it is necessary to make some judgement on the way it affects the segmenting of the picture. If we consider a segmentation successful on the grounds that it is better to have more large regions and fewer very small ones without involving high level knowledge, this method can do the job very well. The merging process improved the quality of the final segmentation, reducing the complexity of the segmentation by eliminating many small regions. The merging process utilized both local pixel similarity information (i.e. color) and size of the region to modify the segmentation.

The total number of regions in the original segmented image (image(5.3a)) is 119, after recursive merging the total region number of the new segmented images of image(6.4a) and (6.4d) are reduced to 84 and 73. The segmented images now appears to be much simpler; they contain less noise, more large regions and fewer small regions. This merging method can also be used during relabelling, in which the segmented region that is smaller than a certain region size will be considered as noise and will be relabelled by its neighbour label. But the computational cost is increased.

Although the results have been encouraging, this approach must face the question whether such small regions with values different from their surroundings are objects to be outlined, or noise to be ignored, or part of a neighbour required merging. This is fundamentally impossible to solve without using higher level knowledge about the class of scenes or without a model for the scenes. This problem can be illustrated using the following example: when this method is applied on the image(6.3e) where region size less than 0.09% of the total image area will be merged, 133 segments are found. The new segmented image boundaries (image(6.5a)) now seems more simpler than image(6.3e). But the right eyeball is longer separated out. However based on the simple criterion such

as the smallest size of object of interest and the color difference, further improvement can be obtained in this way.

6.5 Segmentation Errors

Many methods of segmentation have been suggested in the literature. However, discussion of segmentation errors is rare. One example of an error measure for blood cells segmentation is discussed in [Yasnoff et al 77]: the percentage area incorrectly classified can often provide useful information to supplement subjective evaluation. However, it does not seem to agree well with human observation. One reason for this is that the percentage area measure does not take into account the spatial information inherent in an erroneous segmentation, i.e. a misclassified pixel in the center of an object is given the same weight as one at an edge. In fact human perception of these errors is not the same. So an effective error measure should combine the percentage area misclassified with the spatial information inherent in an erroneous segmentation and correlate this error measure with human observation.

Another example is discussed in [Kohler 81]: comparing an automatic segmentation and a corresponding 'correct' segmentation of an image, segmentation errors can be classified into two primary types. First the segmentation contains regions which do not exist in the 'correct' goal segmentation. Second the segmentation misses regions which do appear in the 'correct' segmentation. There is a third type of error which can be considered as a compound of type one and type two errors. For this type of error the regions in the correct segmentation are correctly detected, but not in the correct locations. The major difficulty in this method is that it is not easy to define the general criteria for 'correct' segmentation of an image. Further, these criteria are ambiguous without knowledge of the

goals of the segmentation system.

Since no quantitative criteria have been established for measuring the segmentation error, it is hard to say how good the algorithm is unless some qualitative statements on the advantages and disadvantages of the methods can be made. Besides, due to the different images used for experiments, it is also not easy to compare each algorithm with others. Even if the same image is used to test different methods, it is still not easy to make a conclusion about which method is the best, since part of the result may be better using method A but the rest better using method B.

The solution might be found, if the segmentation task can be defined and limitations such as required processing time, memory, and cost declared. The first requirement can be illustrated using the color chalks (image(4.9a)). Assuming that the task is to segment the image based on color. Image(6.6a) to (6.6c) is the segmentation using histogramming applied on the normalised red, green and blue component images (image(4.9e) to (4.9g)). None of them can do the job properly. Even their resultant segmented image shown in image(6.6d) is also failed. Comparing them with image(4.9k) to (4.9l), We can conclude that in this case the two-dimensional clustering is superior than the one-dimensional histogramming, since either the image(4.9k) or image(4.9l) can segment the image successively.

If all the methods used can produce the same result, which one is the best one will depend on the second requirement. For example, the task for a computer vision system might be to automatically inspect an electric plug in order to make sure that the three wires (earth, neutral, power) are correctly assembled. Based on this requirement any method is acceptable as long as it can separate the wires from each other and the background (i.e., plug) correctly. Then later processing can be used to identify the wire. The segmentation performance on something else is irrelevant e.g., screws, fuse, and the plug itself. The best

method will be highlighted only when the limitation is imposed. For example say there are two methods, A and B, that can do the job properly. The former is faster but requires more equipment (i.e. is more expensive). The latter is slower but the required equipment is simpler (i.e. is cheaper). At that time we still can not make a conclusion about which is the best one. However, if the time required to complete the job is known and only the latter method can also give the result within that period, clearly the better one is the latter.

Segmentation methods are ad hoc in nature. We can not reject any method as an invalid technique purely on the segmentation result alone. We have to consider whether the feature selected as a basis is appropriate or not (section 4.1). For example color information is an effective and essential feature for an application which requires segmenting the wires in the plug. It is because the electric properties of wires are related to color; blue wire marks the neutral, brown wire marks the live, and so on. A textural method is unlikely to do the job effectively. It does not mean textural method is invalid. The only conclusion is that textural method is not appropriate in solving this specific problem. For a given image the points may be scattered or clustered, depending on the feature space used. However, choice of the optimal feature space which can provide distinct clusters for meaningful segmentation is quite difficult and depends on higher level information. The technique generally employed is to use a wide variety of feature spaces to perform image segmentation, the optimal features being those giving the best segmentation for the image. An example using such a technique can be found in [Ohlander et al 78].

Although there are no correct answers to the problem of how good a segmentation should be and the evaluation of segmentation methods is not easy without specific information on the problem to be solved, segmentation algorithms should at least provide a

complete and structured representation of the image. Then features (i.e., segments) and their relationships can easily be manipulated in the higher level processes using external knowledge such as a model and semantic information, since partial segmentation can not provide the required solution. For example, say we want to count the number of cells in an biological image. The solution should be found if we can segment the cells from each other and the background properly. However, if we imagine the image(5.1a) is a satellite picture where the yacht is a spy ship, and our aim is to locate the yacht in the image, and this job is unlikely to be done without higher level processing. First it may need to identify the four main regions: sky, hill, land, and lake. Second the process is concentrated on the lake area since the yacht should sail on the lake. Third the location of the yacht may be found provided the information about its model (such as color and size) are given when there is more than one boat.

Although there are no quantitative methods to evaluate segmentations, additive noise with different magnitudes have been added to the image before applying the clustering method to illustrate its effect on the method and the segmentation. The noise added is normally distributed gaussian noise with zero mean and unit variance. The red and green component image of the color pattern (image(4.2b) and (4.2c)) are used for this experiment. The clusters found in each noised image are illustrated in the LUTs shown in image(6.7a) to (6.7x) with 64 cluster spaces, and also illustrated in figure(6.3). The segmentation are illustrated in image(6.8a) to (6.8x). And some of the noisy images are illustrated in Image(6.9a) to (6.9l).

Since additive gaussian noise is used, the resultant gray level of the noised image will be equal to the summation of the original gray level and the noise. Therefore it is possible for a resultant gray level having a negative value, which is meaningless physically, so that

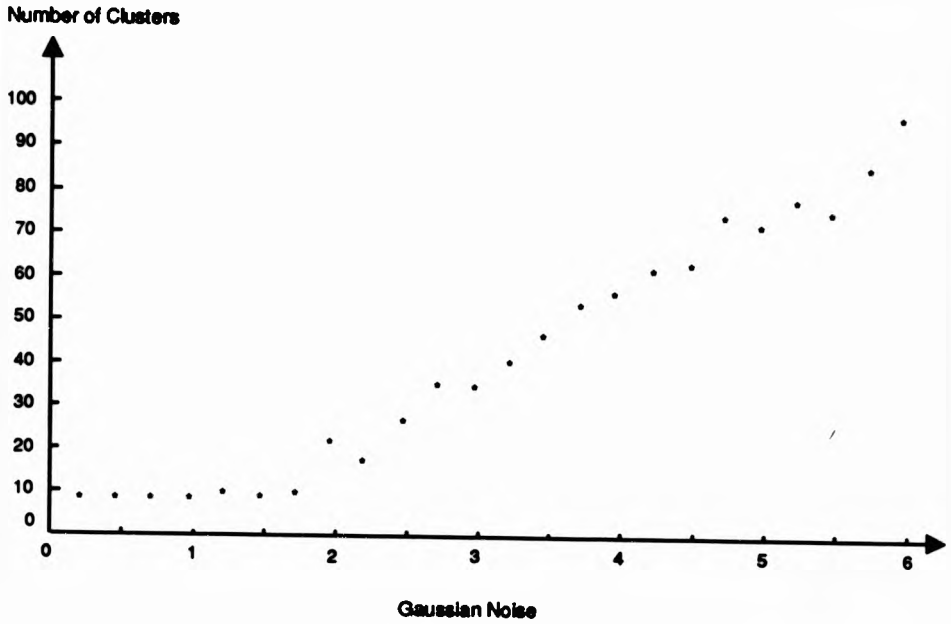


Figure 6.3: Number of clusters plotted against normally distributed gaussian noise with zero mean and unit variance.

a lower bound for the noise image is set to 0. The clipping effect can be seen on the left-hand and top-hand sides of the LUTs.

When the the noise is small and the clusters are well separated, the original clusters may not be affected, so the result might be the same. This can be illustrated in image(6.7a) to (6.7e) and the segmented image (image(6.8a) to (6.8e)). However, if the clusters are very close together, small amount of noise may have an influence. This effect can be seen by comparing image(4.10j), which is the original LUT having 8 clusters without adding noise to the component images. After adding the noise, two of the clusters at the center of the LUTs shown in image(6.7a) to (6.7e) are merged together. And the total cluster

number is reduced to 7. The reason is that this two clusters are very close as shown in image(4.10g), which shows the peaks distribution of the LUT (image(4.10j)).

When the noise is increased, the original cluster will probably be distorted. Some distinct clusters may merge together, and some clusters may split into many smaller separated clusters. Or cluster merging and splitting happen at the same time. This effect can be illustrated in image(6.7f). The central cluster is split and merged with their neighbouring clusters. As a result the region represented by this cluster will be lost (image(6.8f)).

When the noise increase further, the clusters found will become bigger and bigger, since the distributions may drift away from their original position and scatter widely on the cluster space (image(6.7g) to (6.7x)). Moreover, the segmentation will be total unacceptable and fragmentation will occur since the clusters found will be determined by the noise (image(6.8g) to (6.8x)). Under such conditions the clustering method fails, unless the noise can be modelled and be extracted before applying the clustering.

This example also illustrates the difficulty in representing the segmentation result quantitatively. The graph illustrated in figure(6.3) would only tell us that as more noise is added, more clusters result. It cannot give us any idea how the noise affects the segmentation, since the number of clusters do not have direct relation to the segmentation; it is the distributions of the clusters that matters. Assuming a very extreme case happens, and the same number of clusters are found (i.e. a horizontal line in the graph). It is still not possible to draw any conclusion on the segmentation result. The reason is that the same number of clusters does not mean the same pattern of distribution of clusters, so that the segmentation may be seriously affected.

Some points which are worth noting in performing image segmentation are discussed below:

1. The segmentation task: this should be the most important point in the image segmentation problem since we can not have a universal method, which can handle all the problems. Besides, even when the same image is used, different purposes will affect consequent process such as the feature selection, and the segmenting method used. So we must first define our aim.

2. The effective feature: what feature (for example, color, texture, gray level etc.), from which the data is derived, should be used. The decision depends on the specific environment considered (i.e., object characteristics). Usually the criterion of feature selection is often based on either the importance of the feature in characterizing the image or the contribution of the features to the performance of the segmentation results.

3. The feature highlight: what method can highlight the feature, so that the segmentation can be done more effectively. For example, as discussed in section 4.6, different color filters can be used to highlight the object we want, while suppressing the unwanted objects. In this way, the segmentation may be made easier.

4. The segmentation method (for example, edge, region, texture etc.): what is the most appropriate method. Probably this will depend on the feature selected. No definite answer can be given. However, the criterion should be based on the time and complexity. Once the purpose can be fulfilled, the best method should be the simplest method.

5. Result: comparing the experimental result for the task under the given constraints (such as processing time), we can determine whether the method is appropriate or not.

6.6 Summary

Some modifications which might be used to improve the segmentation are discussed in this chapter. These methods include edge points extraction, textured parts extraction and recursive merging.

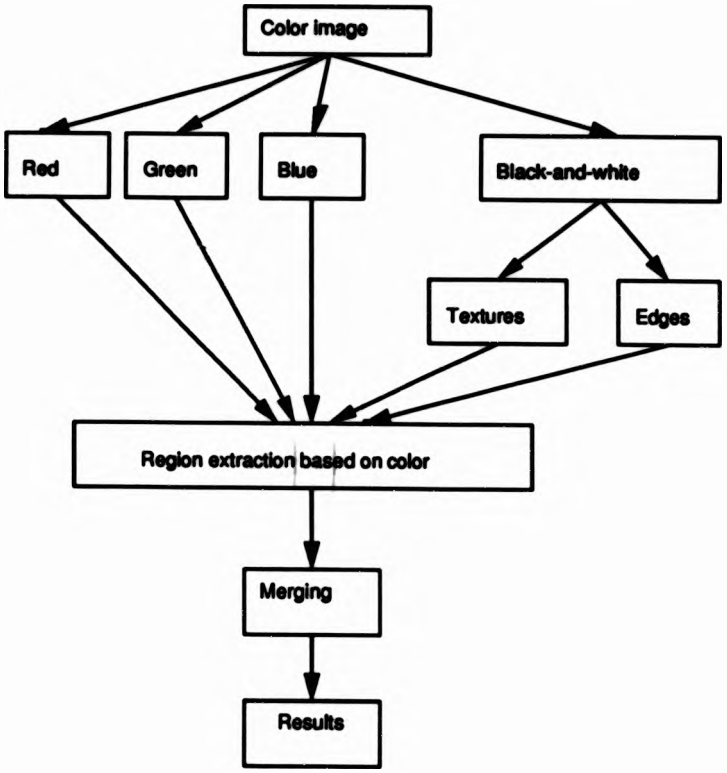


Figure 6.4: An overview of the segmentation algorithm.

These modifications are illustrated in the following block diagram (figure(6.4)). Color image is the original image. Red, green, and blue are the three component images. Black-and-white is the brightness image. Pre-processing can be applied to the black-and-white

image to pre-extract the edge points, which are not used during the segmentation process. These edge points will be replaced by the label of its neighbour pixel during relabelling.

The texture feature of the black-and-white image which is formed using gray-level and local busyness is also examined. Once a cluster which fulfills the specified criterion is found on the texture feature, the corresponding area will be considered as a highly textured area and will be treated as a terminal region. That means it will be saved as a segmented region. Since the effectiveness of the method can not be guaranteed, after completing the segmentation, post-processing extraction is applied on the binary image which requires relabelling. After applying opening and labelling on the binary image, regions whose size is greater than user determined threshold will be considered as a highly textured parts and saved as a single segment.

The reason why the pre-processing texture extraction is still required is that the textured parts of the image will produce scattering in the color feature space. Each isolated point will become a local maximum. This will increase the computational cost. However, if they can be extracted during the pre-processing, the problem can be solved.

After completing the segmentation, post-processing, recursive merging, can be applied to the segmented image to further reduce the fragments under some simple pre-defined criterion. In the case discussed here, the region size and color difference are used. However, other parameters such as average gray level, length of adjacency, texture, and final shape can also be used. Possible improvements in the accuracy of the method can be obtained by using knowledge-based approach. In this case, the regions selected for possible merging will depend on prior knowledge about the structure of each object, driven by means of a model-driven approach. According to this approach, the control structure can derive some hypotheses from the available regions, and test the compliance of the recognized objects

with such hypotheses.

Finally we discuss the problem of segmentation errors and conclude that this is a task dependent problem. It is hard to comment in detail on segmentation results without specifying clearly the task for which the segmentation is being done.

Chapter 7

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

7.1 Conclusions of the Thesis

In this thesis we have considered the application of clustering techniques using watersheds in segmenting color images.

We begin by introducing the problem in image segmentation. The general approach to this problem using edge-based and region-based method is presented. The relationship between these two methods is briefly discussed. The idea of clustering using watersheds and the implementation methods are discussed. The clustering and recursive techniques which have been developed using watersheds is outlined and the effectiveness of the methods are illustrated in the segmentation results. The applications of such methods are highlighted and possible improvements to such methods discussed. Finally there is a brief discussion

in segmentation errors.

In our introduction to watersheds we emphasized that it is a data-driven clustering method for two-dimensional data. Therefore it is very suitable for non-parametric clustering in images since there is usually no prior knowledge of the classes. Clustering using watersheds algorithm applied to image segmentation based on color can be regarded as the multidimensional extension of the concept of thresholding. Since this approach is based on the assumption that different classes of regions (i.e., different colors) of an image are represented by distinct 'modes' in the distribution of the two-dimensional histogram, the technique will fail if this assumption is not true. However, it is not the shortcoming of watersheds, it is coherent with any clustering method based on the same assumption.

An important feature of the two-dimensional histogram clustering using watersheds followed by mapping of cluster labels back to the image is that the two-dimensional histograms provide a global view of the feature data without reflecting the spatial information in the image (i.e. the positional relationships between the pixels) from which they are derived. Thus, formation of clusters in two-dimensional color space does not take into consideration the spatial information inherent in the image. For example, a color random dot image should provide good clusters in color space, but can not provide good region segmentation. It is likely that the segmentation will be improved if the watersheds can be combined with spatial information, although this is not easy. One example which tries to combine the spatial information to help the clustering in feature space was discussed by [Riseman & Arbib 77]. However, this clustering technique is straightforward both to justify, and to implement.

This study looks at two versions of watershed applications. One is direct clustering and the other is recursive clustering.

The direct clustering method is good for images in which prior knowledge can be obtained, such as industrial scenes. In industrial circumstances, we can have prior knowledge about the scene, such as what kinds of color will be there. Besides, the range of the color is usually restricted, and the objects rarely have much variation in color over their surface. So we can easily select the most appropriate LUT (i.e. color features) to do the clustering.

Moreover, we propose using normalised color in place of red, green, and blue component color. The main reason is that red, green, and blue are strongly influenced by intensity. Thus, spurious segmentations tend to occur because of the difference in intensities. Normalised color which can eliminate the factor of intensity embedded in the component color is ideally suited to segmentation. The effectiveness of the method relies on the reduction of fragmentation caused by the variation in brightness in color images.

Recursive clustering is good for images about which we do not have any prior knowledge. Clusters which are distinct in the three-dimensional color space may overlap in the two-dimensional color space. Combining the segments found in each two-dimensional color space (as discussed in chapter 4) is a remedy for solving this problem. However, fragmentation due to multiclass segmentation cannot be avoided if we do not have any prior knowledge about the image. In recursive clustering, only one cluster is selected at a time, so that such fragmentation does not occur. Besides, recursive application can remove the largest peaks, allowing less noticeable peaks to become visible; the reduction in the number of peaks can also reduce their mutual interference. Therefore the problem that small clusters are often not visible when they occur inside large regions can be avoided. Furthermore, peaks may come from different parts of the image, and then overlap. Recursive clustering will only consider smaller portions of the image at any one time.

One major limitation on the method is that the segmentation process takes a lot of

time due to its recursive nature. This can be substantially improved by employing special hardware which can perform high-speed parallel computation.

Both methods do provide an unbiased aid to classification of the image, since they do not require the input of any prior information, with the exception of a few assumptions that need to be made about the initial image. These are the user-determined constants.

How the developed segmentation method can be used for image compression is presented. Both methods of regional and boundary have been discussed and illustrated using the reconstructed image and the segmented image boundaries. Besides, there is a discussion showing how the pyramidal data structure can be used in clustering for coarse segmentation, in which the important image regions based on the significance in the two-dimensional feature (color) spaces are extracted.

The rest of the thesis is a study of improvements to these methods used in segmentation. Edge extraction: pre-processing can be applied to the black-and-white image to pre-extract the edge points, which are not used during the segmentation process. These edge points will be replaced by the label of its neighbour pixel during relabelling. This method is good when it is applied to the man made objects because edges are more apparent than is usually the case with natural scenes. Usually these edges will not affect the segmentation result seriously since their regions are some small broken lines, and can easily be thresholded. However, their existence will increase the computational cost. So it may be wise to pre-extract them before starting the segmentation.

Texture extraction: both clustering methods above do not cope well with the textured parts of a image, since these parts will produce scattering in the two-dimensional color space. As a result fragmentation will occur. To overcome this problem we propose the texture extraction methods. Such methods can be divided into two areas: one is a pre-

processing method and the other is post-processing method. Since the effectiveness of the former method can not be guaranteed, after completing the segmentation the latter method is applied to the binary image which requires relabelling. The latter method is very effective because it overcomes the well known problem in clustering using color that the textured parts are apt to be divided into a lot of tiny, meaningless regions; further, the large parts requiring relabelling are combinations of those parts.

The reason why the textured parts of an image will produce scattering in the color feature space is that each isolated point will become a local maximum. This will increase the computational cost. However, if textured parts can be extracted during the pre-processing, the problem can be solved. It seems that when both methods are applied together, the problems caused by the textured parts seem to have been solved. The penalty is higher computational cost. However, it is hard to have an effective texture extraction method without involving some prior knowledge about the image.

Merging: merging can be applied to the segmented image to further reduce the fragments under some simple pre-defined criterion. In this case, the region size and color difference are used. However, other parameters such as average gray level, length of adjacency, texture, and final shape can also be used. Since these methods still depend on the local properties, the improvements are limited to having more larger regions in the segmented image than that without merging. Further improvement can only be obtained if prior knowledge about the image is available, such as the shape of the objects. That means the merging algorithm becomes model-driven.

Although there are no formal methods to assess the segmentation results, based on the author's subjective evaluation, the algorithm developed here represents some success in partial segmentation based on the use of clustering techniques of watersheds in color

space. The performance of the methods is satisfactory within the physical limitations such as 64 gray-level resolution, the varying noise of the output of the camera and also the noise intrinsic in the image. These clustering techniques can be used in a wide variety of applications. The current concern is with color images but the technique is only restricted to images whose features can be represented in two or more dimensions. Then the catchment basin or the cluster as shown in figure(3.14) can be located and mapped back to the original image to do the segmentation. For example the feature space used for clustering based on texture as shown in section 6.3.1 is a two-dimensional histogram formed by using the gray-level values and the local busyness.

As the concluding remarks to this thesis I remark that:

1. Image segmentation tasks are problem dependent, so that no single method is sufficient for all applications, and no optimal method exists for all images. The method used implicitly reflects the user knowledge and expectations about the scene domain. Therefore the selection of an appropriate method is far easier if prior knowledge about the image is available and the aim is specified. Explicit representation of such knowledge may also enable a user to make different assumptions in different cases and improve various constraints to make the selected segmentation algorithm proceed more effectively.

2. It is almost impossible to implement a practically useful complete segmentation technique based only on local predicates without including any external knowledge in the segmentation process, such as semantic knowledge about the nature of the scene being analysed. This knowledge is used to aid the segmentation of the image into regions that correspond roughly to the objects expected in the scene. This problem can be illustrated using letter 'A' and 'i' as shown in figure(5.2). Although

the letter 'I' looks simpler than 'A', it can not be assigned as one segment (i.e. one object) since it is not connected.

One of the reasons is that an image, which can be considered as a two-dimensional signal varying in space and gray level (as discussed in section 1.1), can represent an infinite number of possibilities [Fu & Mui 81]. So we can not built a general image understanding system based on local properties unless we know exactly what kind of knowledge is required. Otherwise, it would require the storage of a vast amount of knowledge due to the infinite number of possibilities. However, successful incorporation of knowledge in segmentation is much simpler when the problem is limited to one specific application. For example an industrial vision system can be used for component recognition and a prior knowledge about the possible components is available.

7.2 Suggestions for Future Research

Some possible methods which could be used to improve the segmentation results and are worthy for future research are discussed below.

7.2.1 Color Decorrelation and Highlights Separation

It is possible that the color based segmentation could be improved by using color decorrelation. The color images are correlated in that they lie in an area near the diagonal of the cube, and so after projection onto the face of the cube, the clusters which can be located in the cube may disappear due to their overlapping. Decorrelation would expand the distribution into the corners of the R,G,B color space. One possible technique for this is described by Herault [Herault et al 89]. Preprocessing of this type should decrease color

image correlation, and help segmentation.

Highlights are an unsolved problem in segmentation, since the highlighted parts of an object do have significantly different brightness values than their surroundings although they lie on the same surface. So it will upset our definition in defining a region and an edge. Worse, it may possibly induce color clipping because of the limited dynamic range of the camera. The simple way to solve this problem is, of course, by moving the position of objects or/and the position of the lighting, so an almost even illumination to the scene and reflection from the objects can be obtained.

However, this problem can also be solved by firstly obtaining a physics-based color reflection model for the highlight and then separating it from the object color as described by [Klinker el at 87, 88, 90]. In their papers, they noted that the color of every pixel from an object can be considered as a linear combination of the highlight color caused by the reflection of the material surface and the object color caused by the reflection in the material body.

When light hits the surface of a dielectric material, the change in refraction indices causes both reflection and refraction. The process of light reflecting back into air at the material surface is referred as surface reflection. It generally appears as highlight on an object. But not all incident light is reflected back at the material surface: the refracted light penetrates into the material body, travelling through the medium and hitting pigments from time to time. The light is partially or entirely absorbed and changed into heat at some wavelengths. The residue of it is scattered by the pigments. Finally some of the light exits from the material body back into the air. This reflection process is referred as body reflection.

The color of every pixel from an object can then be separated into a matte object

component and a highlight component using the Dichromatic Reflection Model [Klinker el at 87, 88]. This generates two intrinsic images: the intrinsic matte object image of the scene showing the scene without highlights, and the intrinsic highlight image of the scene showing only the highlights of the scene. The matte object image may then be used for segmentation.

7.2.2 Planning and Focussing

Planning can be used to improve the speed of the segmentation. Segmentation of large images is always very time-consuming, especially when the method is recursive in nature. To solve this problem, planning has been used. The planning procedure performs the segmentation of the reduced images and uses this segmentation as a plan for the final segmentation of the full size images. For example, planning used by [Ohlander et al 78] involves creating a reduced version of the image, in which the pixel value is the average of the value in the original image. The threshold, which is determined using the same segmentation procedure on the reduced image, are applied to the original image to produce the final segmentation. This technique requires less computational time. However, details tend to disappear. This will not matter if we are looking for the major feature or the approximate nature of the major feature of the image.

Another method, focussing, is recommended by [Shafer 80] to improve the speed of segmentation. The method involves extracting the original image around objects of interest. The advantage is that, since extracted image contains fewer objects, the costly computational problem can be minimized. However, this technique can only be used when there is some specific knowledge, such as semantic analysis, to identify the 'objects of interest'.

Using focussing in planning seems to be an interesting topic in segmentation. The

idea seems very simple. Firstly, the original image is reduced in size to form a pyramid. Secondly, searching is applied on the reduced image. If the objects of interest are found, searching on this level is stopped, and one proceeds to the original image to do the segmentation. However, if the objects of interest cannot be found, the searching will continue from the topmost level (the most coarse image) until the base (the original image). Reduced images are used for searching for the object and only part of the original image, the vicinity area of the object, is used to do the segmentation. The computation time should be reduced.

However, if this approach were adopted, it would make the central problem for image segmentation arranging for the right piece of specialized knowledge to be made available at the appropriate time during segmentation. The reason is that segmentation may make an object surface separate from the background or from other object surfaces but it does not do the job of detecting the presence of such object in the image. This could only be done when the segmentation is presented in a hierarchical manner, in which not only the structure or relationship between the segments or features is defined, but also the order of importance between the features is given, since the objects in an image can be a composition of a number of features and some of them are the key features and the rest are less important. For example, the wheels of a car might be considered as the most important feature in identifying the presence of a car.

The addition of such higher level (structural) information eventually would turn the problem into an artificial intelligent problem: however, the work reported here aims to produce a base, which is data driven, on which higher level processing could be attempted.

Appendix A

Images For Chapter Four

Image(4.1a) Leaves .
Image(4.1b) The segmented image of the leaves.

Image(4.2a) The B/W image of the color pattern.
Image(4.2b) The red component image.
Image(4.2c) The green component image.
Image(4.2d) The blue component image.
Image(4.2e) The segmentation based on R/G images.
Image(4.2f) The segmentation based on R/B images.
Image(4.2g) The segmentation based on G/B images.
Image(4.2h) The resultant segmented image.
Image(4.2i) The regions required relabelling are shown in black.
Image(4.2j) The final segmented image.
Image(4.2k) The segmented image boundaries.
Image(4.2l) The boundaries superimposed on the B/W image.

Image(4.3a) Binary image extracted from image(4.2h) for opening.
Image(4.3b) Image after erosion applied to image(4.3a).
Image(4.3c) Image after dilation applied to image(4.3a).

Image(4.4a) 1st eroded image of image(4.1b).
Image(4.4b) 2nd eroded image of image(4.1b) .
Image(4.4c) 1st dilated image of image(4.4b).
Image(4.4d) 2nd dilated image of image(4.4b).
Image(4.4e) Number of leaves represented by different gray levels.

Image(4.5a) Binary image extracted from image(4.2h) for closing.
Image(4.5b) Image after dilation applied to image(4.5a).
Image(4.5c) Image after erosion applied to image(4.5b).

Image(4.6a) Binary image extracted from image(4.2j) for edge detection.
Image(4.6b) Image after erosion applied to image(4.6a).
Image(4.6c) Image after dilation applied to image(4.6a).
Image(4.6d) Internal edge of image(4.6a).
Image(4.6e) External edge of image(4.6a).
Image(4.6f) True edge of image(4.6a).
Image(4.6g) Edge found using a 3×3 structural element .

Image(4.7a) Binary image extracted from image(5.3a) for edge detection.
Image(4.7b) Image after erosion applied to image(4.7a).
Image(4.7c) Image after dilation applied to image(4.7a).
Image(4.7d) Internal edge of image(4.7a).
Image(4.7e) External edge of image(4.7a).
Image(4.7f) True edge of image(4.7a).

Image(4.8a) The B/W image of the color block worlds.
Image(4.8b) The red component image.
Image(4.8c) The green component image.
Image(4.8d) The blue component image.
Image(4.8e) The normalised red component image.
Image(4.8f) The normalised green component image.
Image(4.8g) The normalised blue component image.
Image(4.8h) The segmentation based on R/G image.
Image(4.8i) The segmentation based on R/B image.
Image(4.8j) The segmentation based on G/B image.
Image(4.8k) The segmentation based on normalised R/G image.
Image(4.8l) The segmentation based on normalised R/B image.
Image(4.8m) The segmentation based on normalised G/B image.
Image(4.8n) Edge computed using Robert cross operator on the B/W image.
Image(4.8o) Edge computed using Sobel operator on the B/W image.
Image(4.8p) Edge computed using Lapacian operator on the B/W image.

Image(4.9a) The B/W image of the color chalks.
Image(4.9b) The red component image.
Image(4.9c) The green component image.
Image(4.9d) The blue component image.
Image(4.9e) The normalised red component image.
Image(4.9f) The normalised green component image.
Image(4.9g) The normalised blue component image.
Image(4.9h) The segmentation based on R/G image.
Image(4.9i) The segmentation based on R/B image.
Image(4.9j) The segmentation based on G/B image.
Image(4.9k) The segmentation based on normalised R/G image.
Image(4.9l) The segmentation based on normalised R/B image.
Image(4.9m) The segmentation based on normalised G/B image.
Image(4.9n) Edge computed using Robert cross operator on the B/W image.
Image(4.9o) Edge computed using Sobel operator on the B/W image.
Image(4.9p) Edge computed using Lapacian operator on the B/W image.

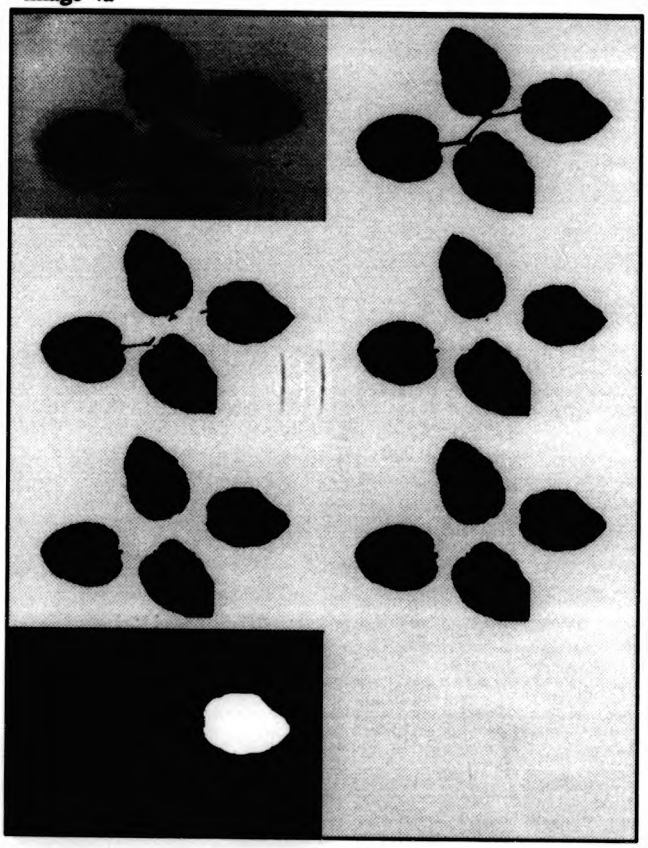
Image(4.10a) Two-dimensional histogram formed by R/G component images.
Image(4.10b) Two-dimensional histogram formed by R/B component images.
Image(4.10c) Two-dimensional histogram formed by G/B component images.
Image(4.10d) Smoothing using 3 × 3 window in image(4.10a).
Image(4.10e) Smoothing using 3 × 3 window in image(4.10b).
Image(4.10f) Smoothing using 3 × 3 window in image(4.10c).
Image(4.10g) Peaks found in image(4.10d).
Image(4.10h) Peaks found in image(4.10e).
Image(4.10i) Peaks found in image(4.10f).
Image(4.10j) LUT of R/G component images.
Image(4.10k) LUT of R/B component images.
Image(4.10l) LUT of G/B component images.

Image(4.11a) Two-dimensional histogram formed by image(4.9e) & (4.9g).
Image(4.11b) Peaks found in image(4.11a).
Image(4.11c) Smoothing using 3 × 3 window in image(4.11a).
Image(4.11d) Peaks found in image(4.11c).
Image(4.11e) Smoothing using 5 × 5 window in image(4.11a).
Image(4.11f) Peaks found in image(4.11e).
Image(4.11g) Smoothing using 7 × 7 window in image(4.11a).
Image(4.11h) Peaks found in image(4.11g).
Image(4.11i) Smoothing using 9 × 9 window in image(4.11a).
Image(4.11j) Peaks found in image(4.11i).

Image(4.12a) Segmentation without smoothing.
Image(4.12b) Segmentation using 3 × 3 smoothing window (duplication of image(4.9l)).
Image(4.12c) Segmentation using 5 × 5 smoothing window.
Image(4.12d) Segmentation using 7 × 7 smoothing window.
Image(4.12e) Segmentation using 9 × 9 smoothing window.

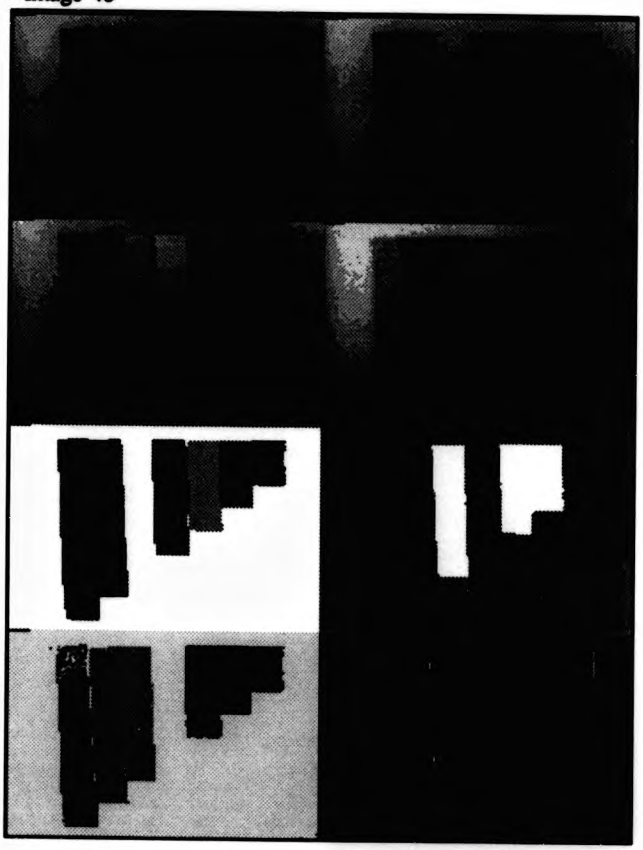
4.1a	4.1b
4.4a	4.4b
4.4c	4.4d
4.4e	

image-4a



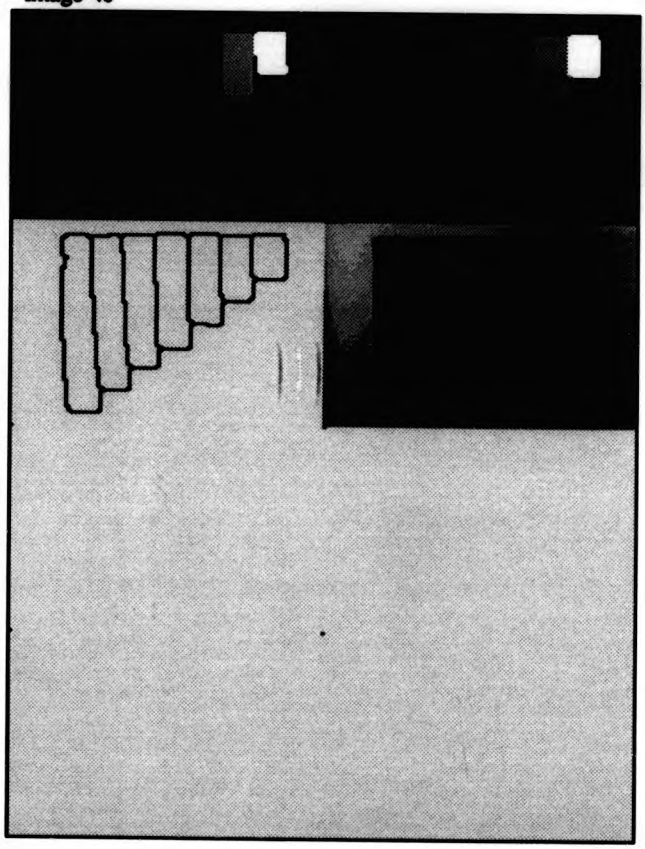
4.2a	4.2b
4.2c	4.2d
4.2e	4.2f
4.2g	4.2h

image-4b



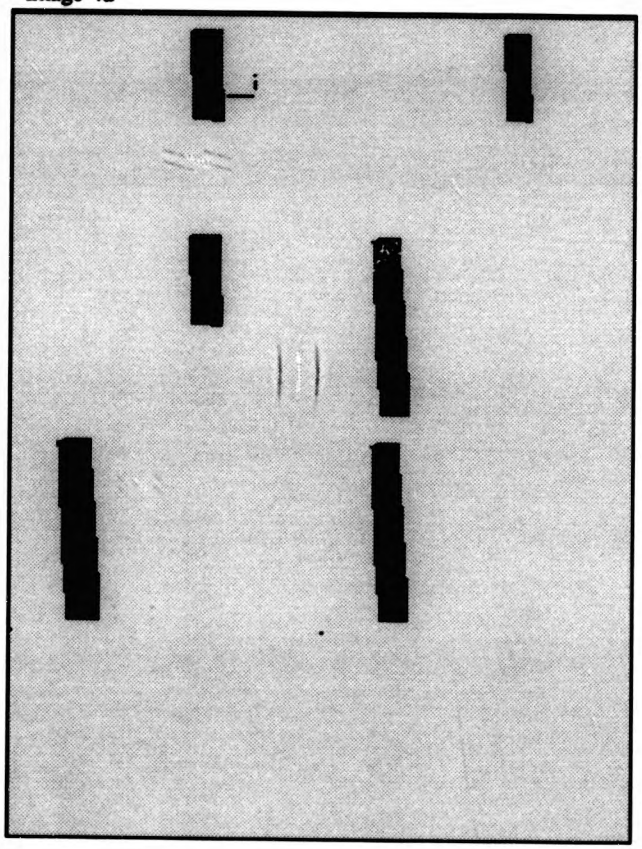
4.2i	4.2j
4.2k	4.2l

image-4c



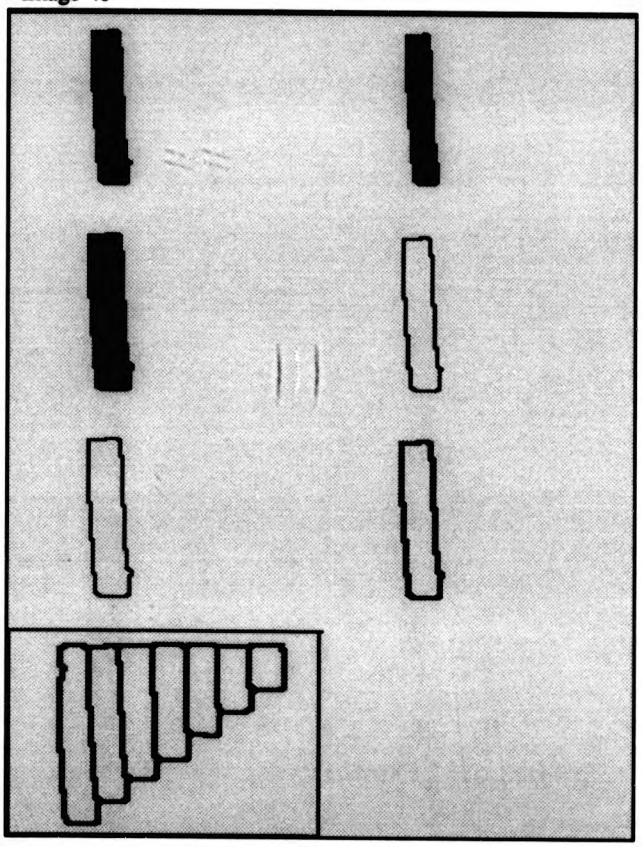
4.3a	4.3b
4.3c	4.5a
4.5b	4.5c

image-4d



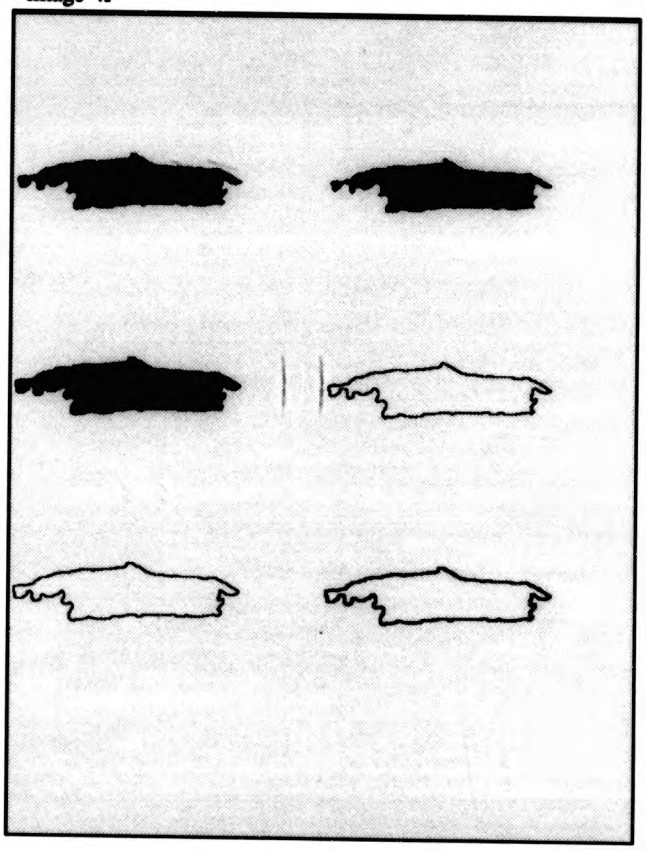
4.6a	4.6b
4.6c	4.6d
4.6e	4.6f
4.6g	

image-4e



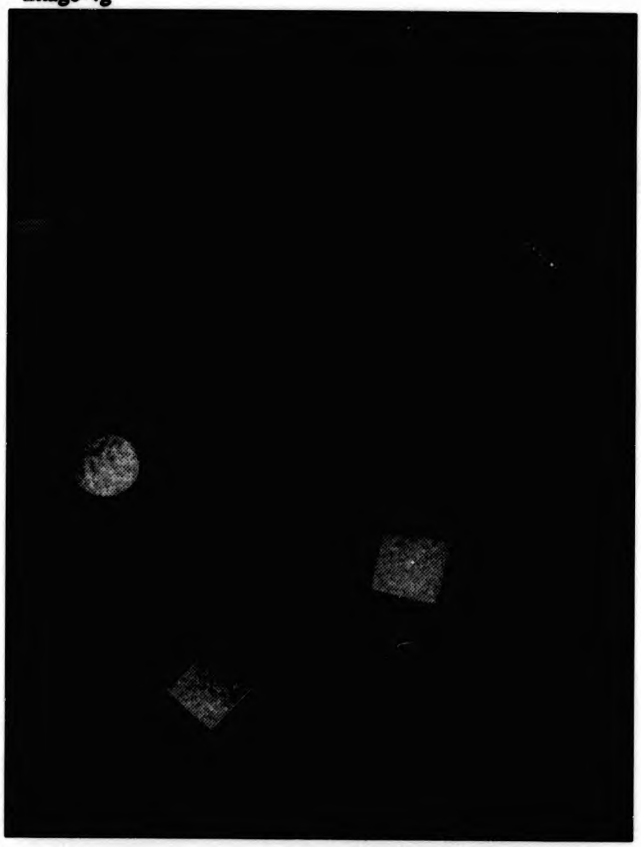
4.7a	4.7b
4.7c	4.7d
4.7e	4.7f

image-4f



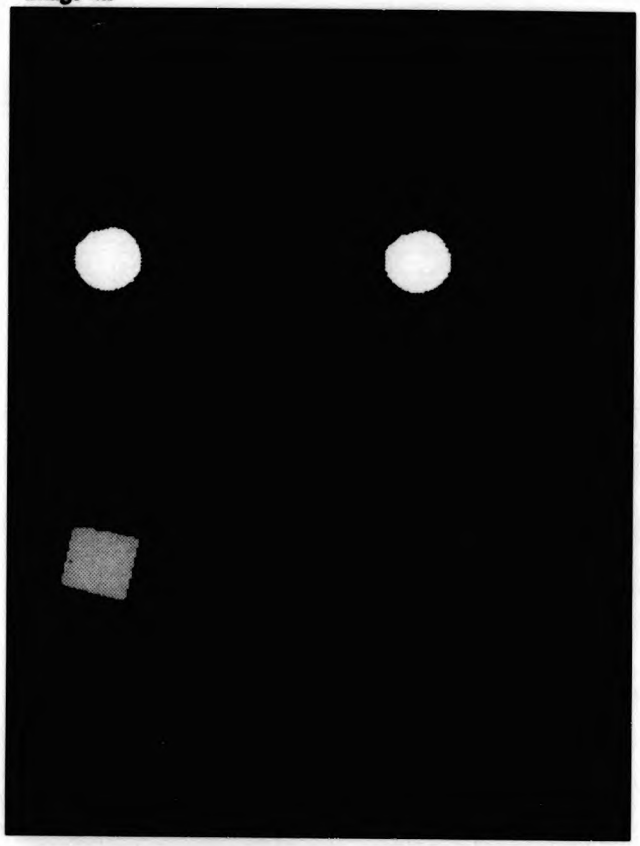
4.8a	4.8b
4.8c	4.8d
4.8e	4.8f
4.8g	4.8h

image-4g



4.8i	4.8j
4.8k	4.8l
4.8m	4.8n
4.8o	4.8p

image-4h

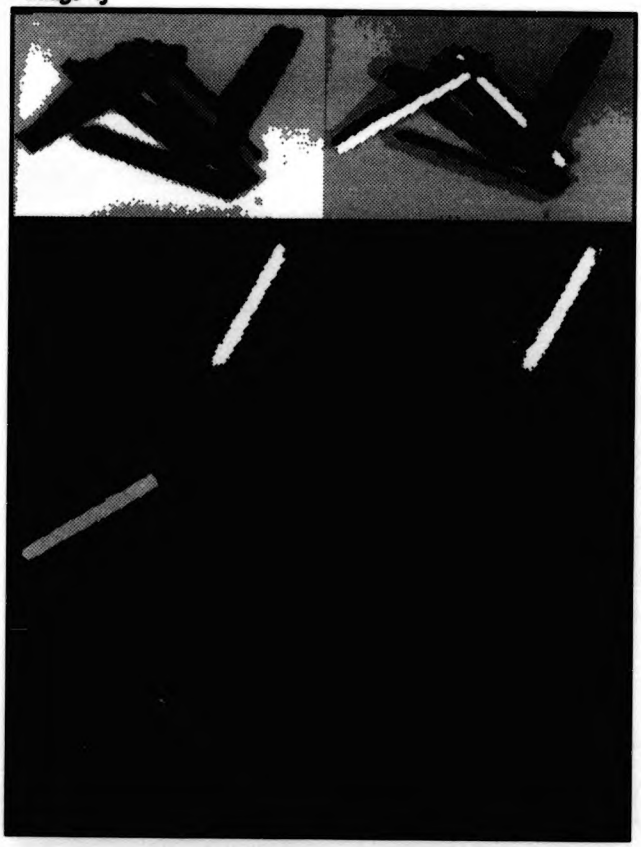


4.9a	4.9b
4.9c	4.9d
4.9e	4.9f
4.9g	4.9h

image-4i

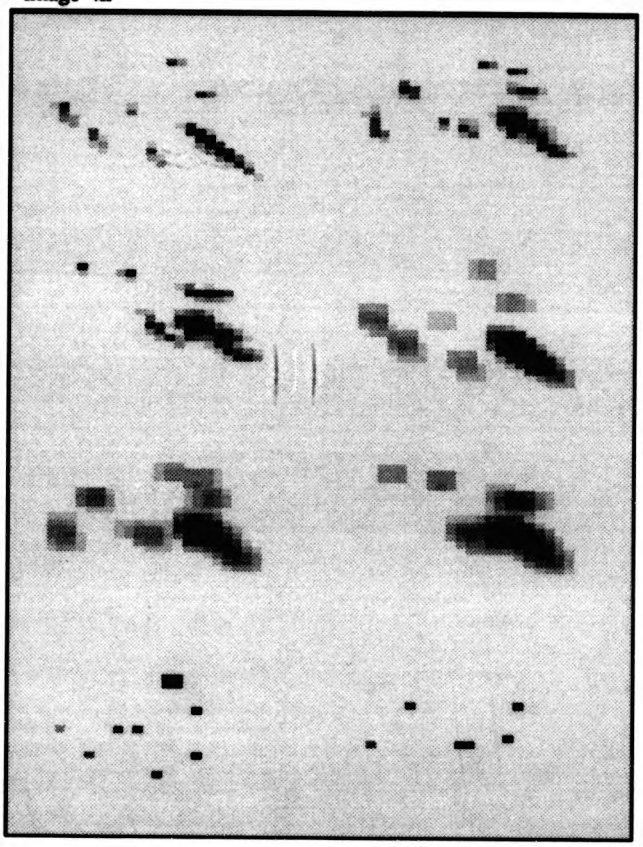
4.9i	4.9j
4.9k	4.9l
4.9m	4.9n
4.9o	4.9p

image-4j



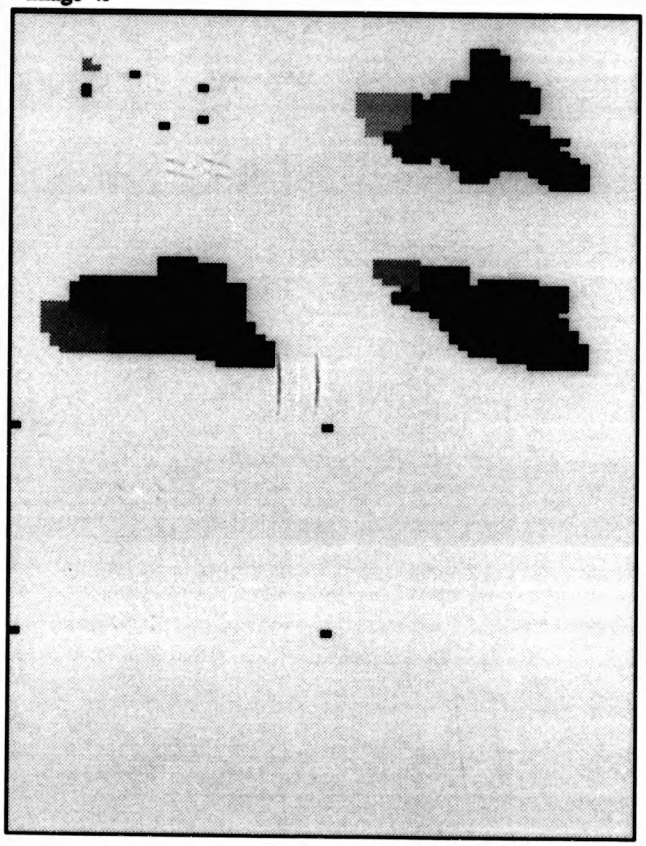
4.10a	4.10b
4.10c	4.10d
4.10e	4.10f
4.10g	4.10h

image-4k



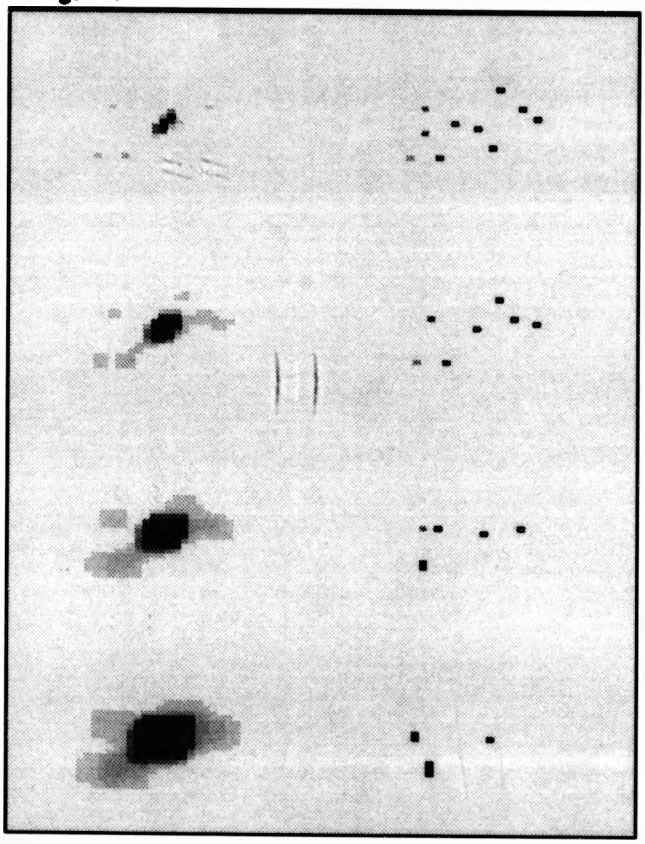
4.10i	4.10j
4.10k	4.10l

image-4l



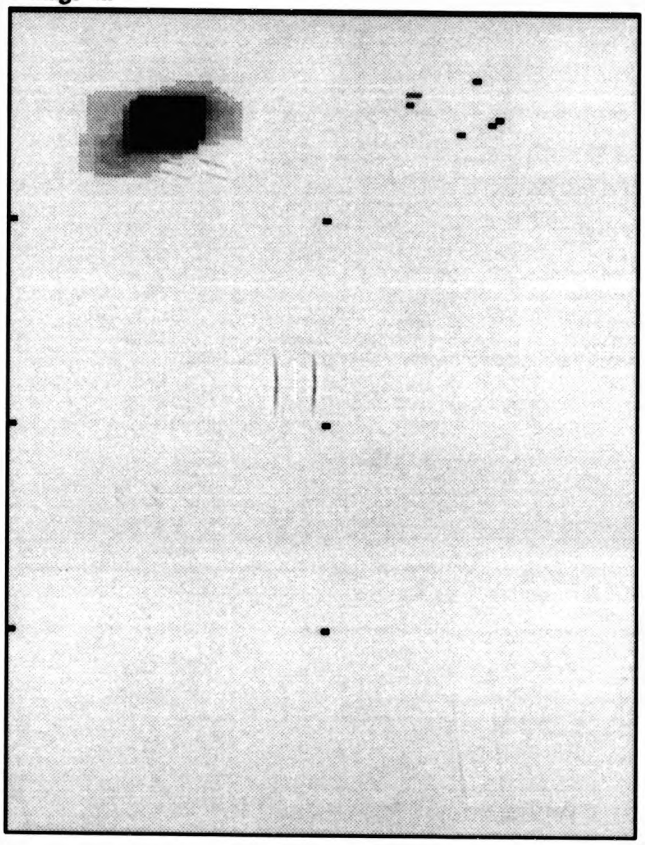
4.11a	4.11b
4.11c	4.11d
4.11e	4.11f
4.11g	4.11h

image-4m



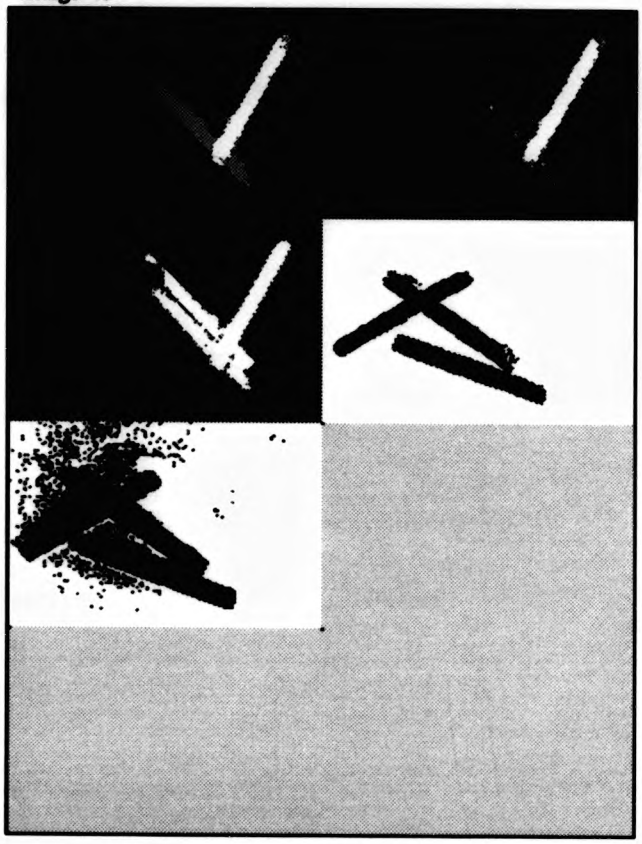
4.11i	4.11j

image-4n



4.12a	4.12b
4.12c	4.12d
4.12e	

image-4o



Appendix B

Images For Chapter Five

Image(5.1a) The B/W image of the landscape.
Image(5.1b) The red component image.
Image(5.1c) The green component image.
Image(5.1d) The blue component image.
Image(5.1e) The segmentation based on R/G images.
Image(5.1f) The segmentation based on R/B images.
Image(5.1g) The segmentation based on G/B images.
Image(5.1h) The resultant segmented image.

Image(5.2a) In 1st clustering, three regions are found in the color pattern.
Image(5.2b) In 2nd clustering, the largest region is divided into two regions.
Image(5.2c) In 3rd clustering, background is saved as a complete segment.
Image(5.2d) Sequence of regions extraction.
Image(5.2e) "
Image(5.2f) "
Image(5.2g) "
Image(5.2h) "
Image(5.2i) "
Image(5.2j) " and the regions required relabelling are shown in black.
Image(5.2k) The segmented image.
Image(5.2l) The segmented image boundaries.
Image(5.2m) The boundaries superimposed on the B/W image.

Image(5.3a) The segmented image of the landscape.
Image(5.3b) The segmented image boundaries.
Image(5.3c) The boundaries superimposed on the B/W image.

Image(5.4a) The B/W image of the road.
Image(5.4b) The red component image.
Image(5.4c) The green component image.
Image(5.4d) The blue component image.
Image(5.4e) The normalised red component image.
Image(5.4f) The normalised green component image.
Image(5.4g) The normalised blue component image.
Image(5.4h) The segmented image.
Image(5.4i) The segmented image boundaries.
Image(5.4j) The boundaries superimposed on the B/W image.
Image(5.4k) The segmented image based on normalised color.
Image(5.4l) The segmented image boundaries based on normalised color.
Image(5.4m) The boundaries based on normalised color superimposed on the B/W image.

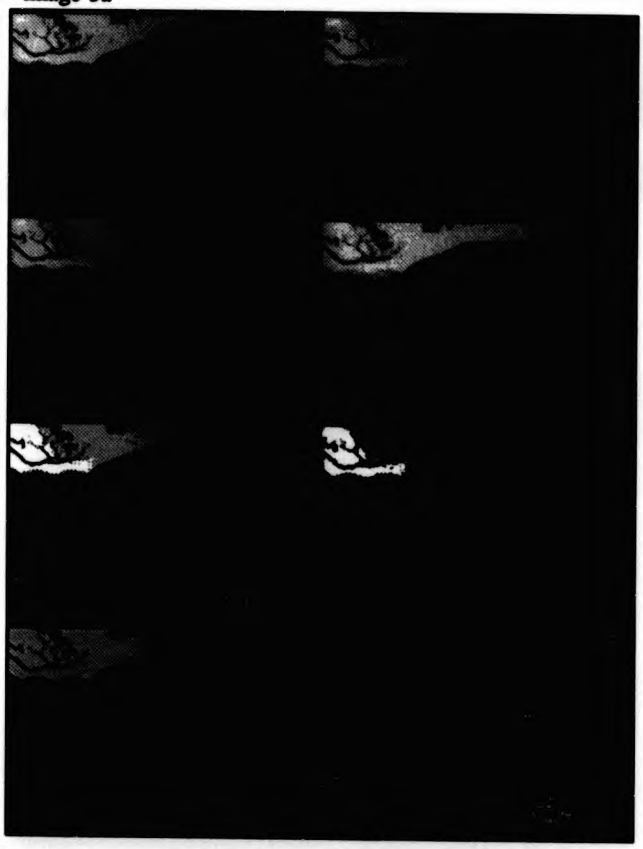
Image(5.5a) The segmented image of color building blocks.
Image(5.5b) The segmented image boundaries.
Image(5.5c) The boundaries superimposed on the B/W image.
Image(5.5d) Sequence of regions extracted based on normalised color.
Image(5.5e) "
Image(5.5f) "
Image(5.5g) "
Image(5.5h) The segmented image based on normalised color .
Image(5.5i) The segmented image boundaries based on normalised color.
Image(5.5j) The boundaries based on normalised color superimposed on the B/W image.

Image(5.6a) The segmented image of color chalks.
Image(5.6b) The segmented image boundaries.
Image(5.6c) The boundaries superimposed on the B/W image.
Image(5.6d) Sequence of regions extracted based on normalised color.
Image(5.6e) "
Image(5.6f) "
Image(5.6g) "
Image(5.6h) "
Image(5.6i) "
Image(5.6j) "
Image(5.6k) "
Image(5.6l) "
Image(5.6m) "
Image(5.6n) The segmented image based on normalised color.
Image(5.6o) The segmented image boundaries based on normalised color.
Image(5.6p) The boundaries based on normalised color superimposed on the B/W image.

Image(5.7a) The B/W image of the mandrill.
Image(5.7b) The red component image.
Image(5.7c) The green component image.
Image(5.7d) The blue component image.
Image(5.7e) The reconstructed image based on image(6.3e).
Image(5.7f) The segmented image using level 1 of the pyramid.
Image(5.7g) The segmented image using level 2 of the pyramid.
Image(5.7h) The segmented image using level 3 of the pyramid.
Image(5.7i) The segmented image boundaries based on image(5.7f).
Image(5.7j) The segmented image boundaries based on image(5.7g).
Image(5.7k) The segmented image boundaries based on image(5.7h).

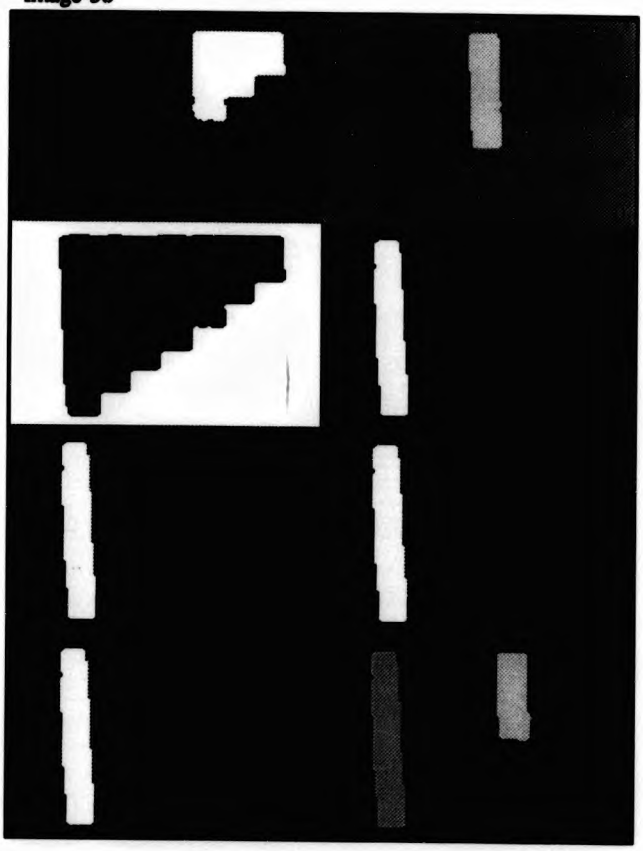
5.1a	5.1b
5.1c	5.1d
5.1e	5.1f
5.1g	5.1h

image-5a



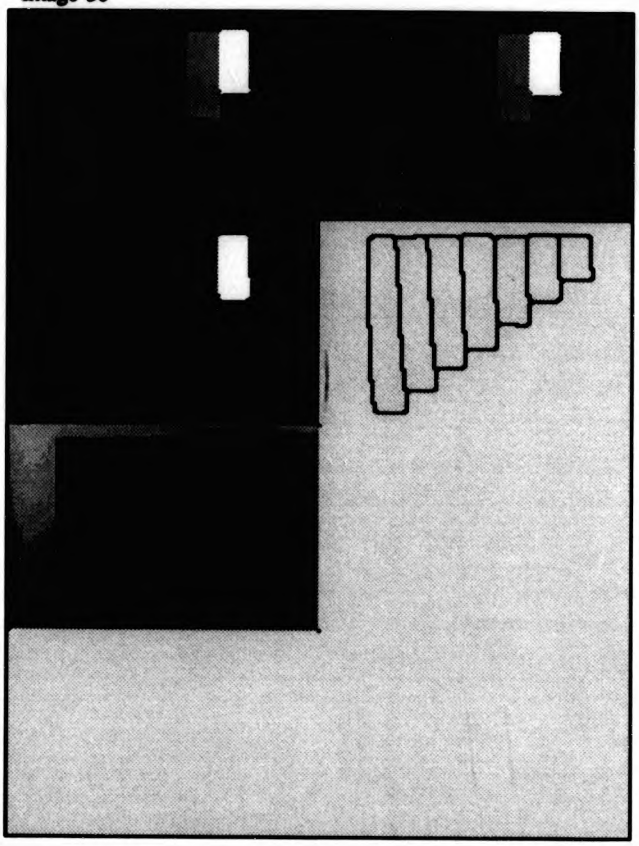
5.2a	5.2b
5.2c	5.2d
5.2e	5.2f
5.2g	5.2h

image-5b



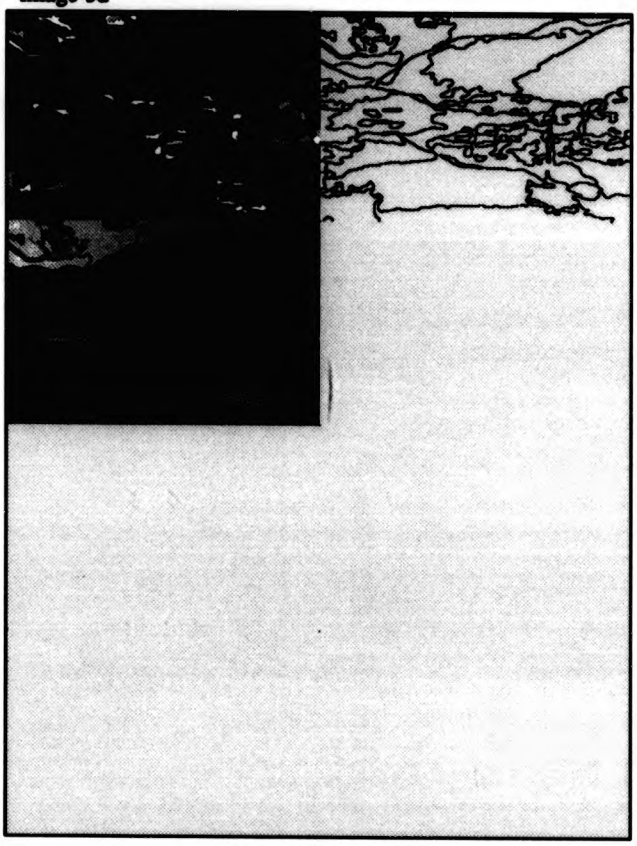
5.2i	5.2j
5.2k	5.2l
5.2m	

image-5c



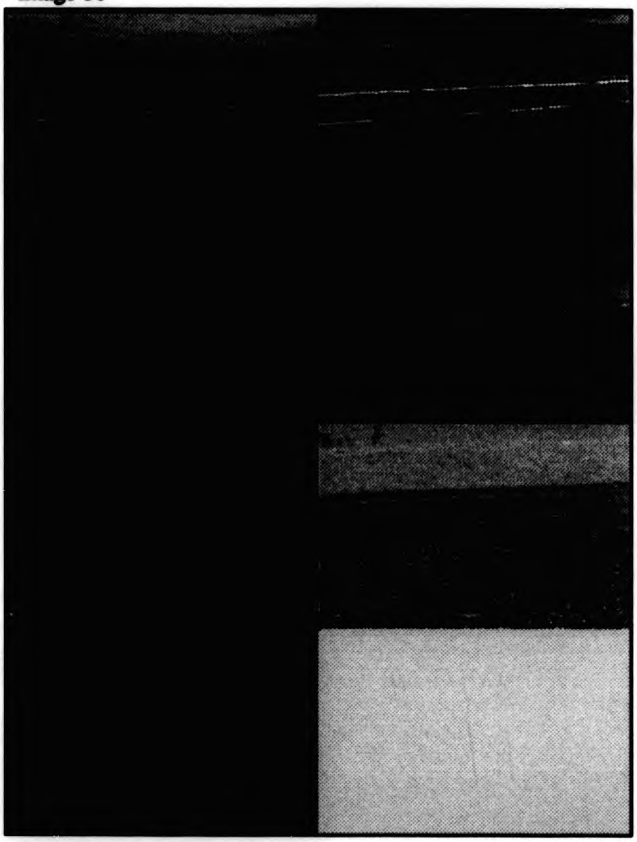
5.3a	5.3b
5.3c	

image-5d



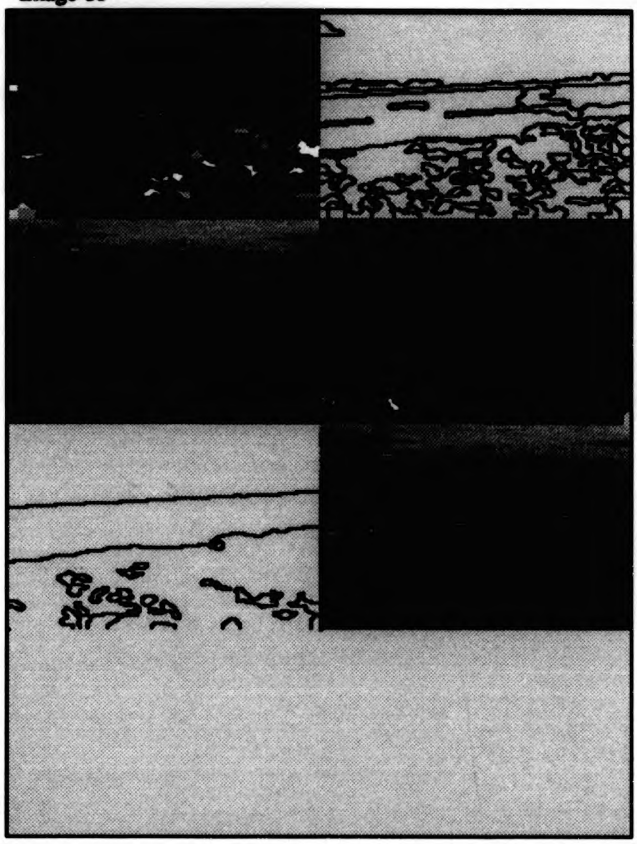
5.4a	5.4b
5.4c	5.4d
5.4e	5.4f
5.4g	

image-5e



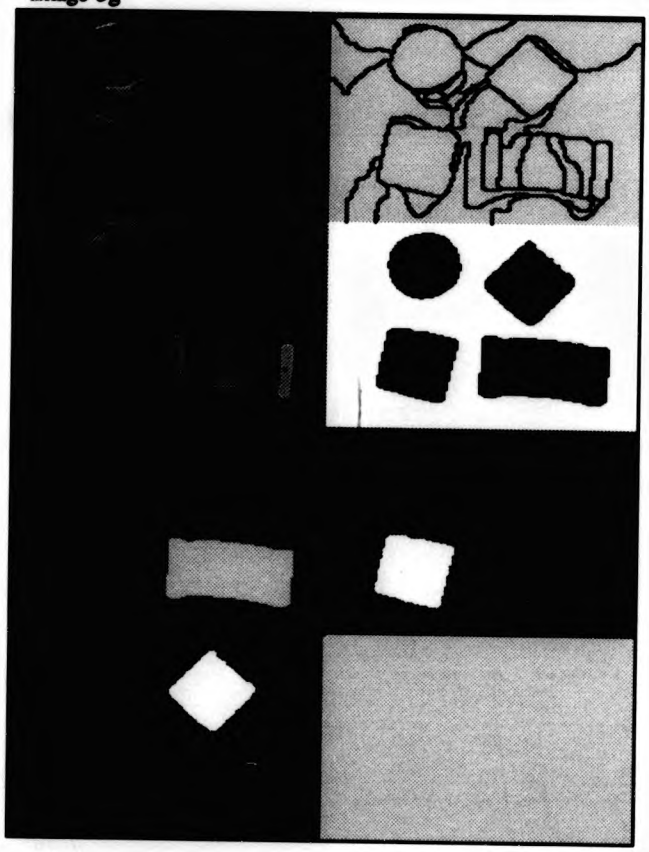
5.4h	5.4i
5.4j	5.4k
5.4l	5.4m

image-5f



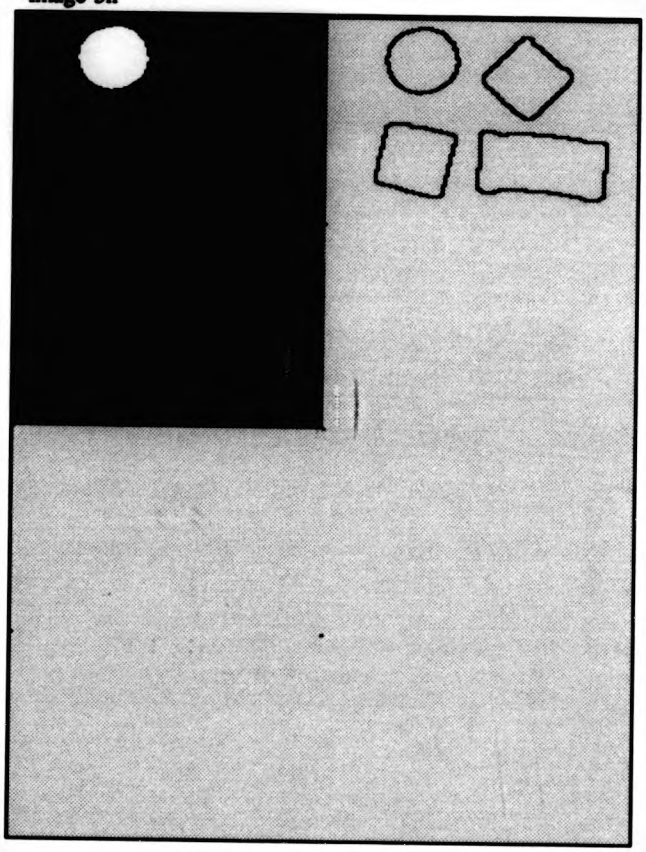
5.5a	5.5b
5.5c	5.5d
5.5e	5.5f
5.5g	

image-5g



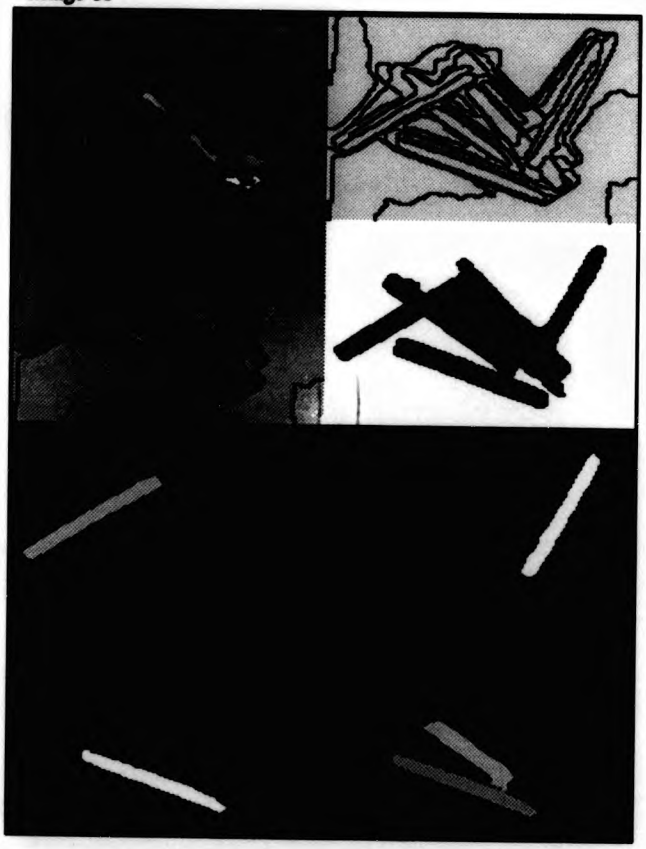
5.5h	5.5i
5.5j	

image-5h



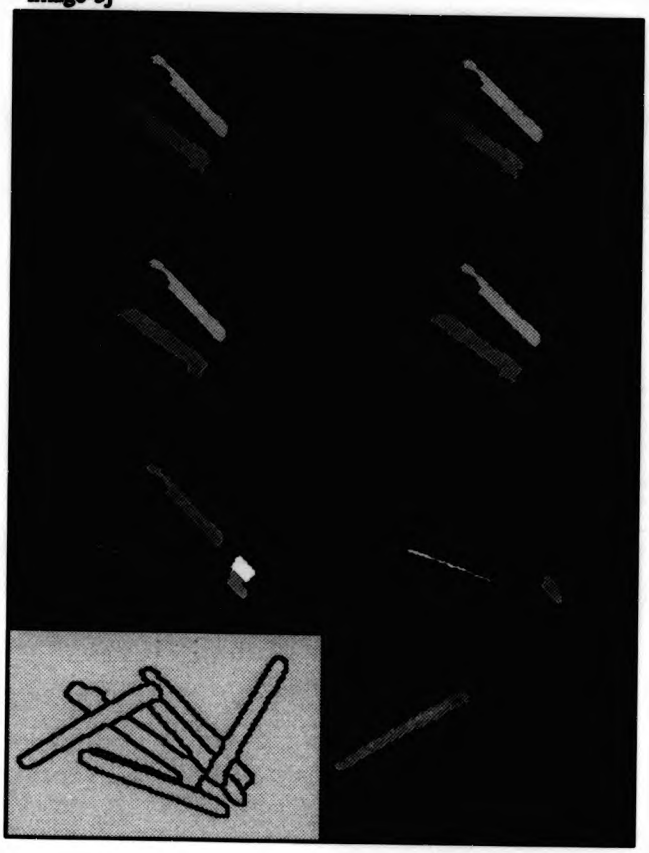
5.6a	5.6b
5.6c	5.6d
5.6e	5,6f
5.6g	5.6h

image-5i



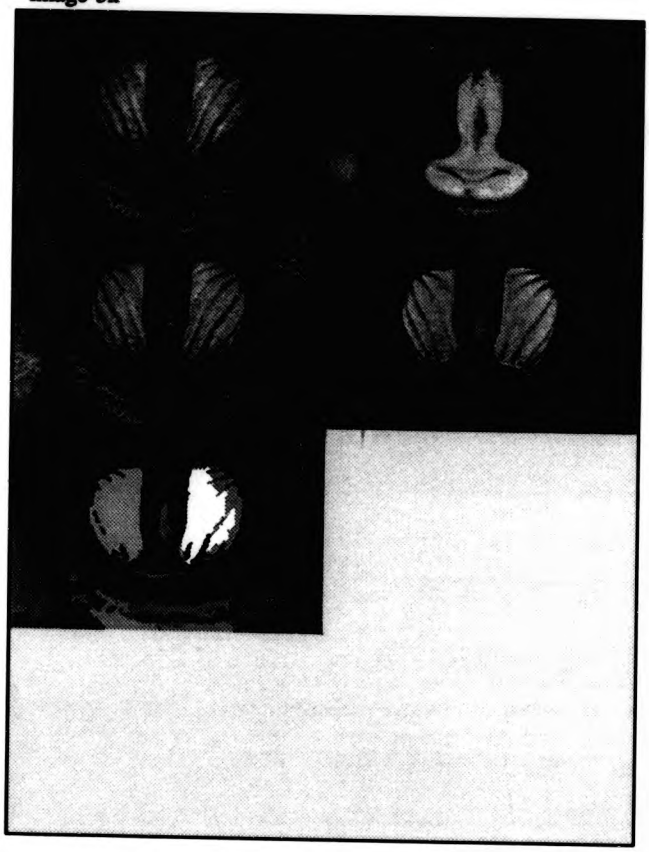
5.6i	5.6j
5.6k	5.6l
5.6m	5.6n
5.6o	5.6p

image-5j



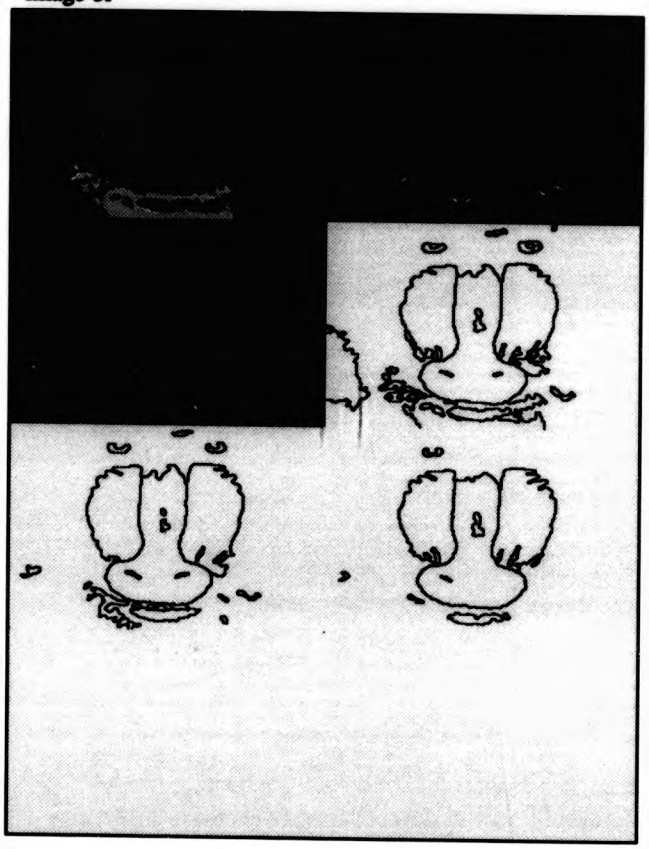
5.7a	5.7b
5.7c	5.7d
5.7e	

image-5k



5.7f	5.7g
5.7h	5.7i
5.7j	5.7k

image-5l



Appendix C

Images For Chapter Six

Image(6.1a) Gradient image computed using Robert's cross operator on image(4.2a).
Image(6.1b) The edge image.
Image(6.1c) The final segmented image.
Image(6.1d) The segmented image boundaries.
Image(6.1e) The segmented image boundaries superimposed on image(4.2a).

Image(6.2a) Gradient image computed using Robert's cross operator on image(5.4a).
Image(6.2b) The segmented image.
Image(6.2c) The segmented image boundaries.
Image(6.2d) The segmented image boundaries superimposed on image(5.4a).

Image(6.3a) Regions require relabelling.
Image(6.3b) Regions extracted by a thresholding of 20 pixels.
Image(6.3c) Regions extracted by a thresholding of 18 pixels.
Image(6.3d) Segmented image boundaries using a thresholding of 20 pixels.
Image(6.3e) Segmented image boundaries using a thresholding of 18 pixels.
Image(6.3f) Segmented image boundaries (image(6.3e)) superimposed on image(5.7a).
Image(6.3g) Edge computed using Robert cross operator on the B/W image(5.7a).
Image(6.3h) Edge computed using Sobel operator on the B/W image(5.7a).
Image(6.3i) Edge computed using Lapacian operator on the B/W image(5.7a).

Image(6.4a) The merged image with region smaller than 0.06%.
Image(6.4b) The merged image boundaries.
Image(6.4c) The merged image boundaries superimposed on image(5.1a).
Image(6.4d) The merged image with region smaller than 0.09%.
Image(6.4e) The merged image boundaries.
Image(6.4f) The merged image boundaries superimposed on image(5.1a).

Image(6.5a) The merged image boundaries with region smaller than 0.09%.

Image(6.6a) The segmentation using histogramming applied to image(4.9e).
Image(6.6b) The segmentation using histogramming applied to image(4.9f).
Image(6.6c) The segmentation using histogramming applied to image(4.9g).
Image(6.6d) The resultant segmented image based on image(6.6a) to (6.6c).

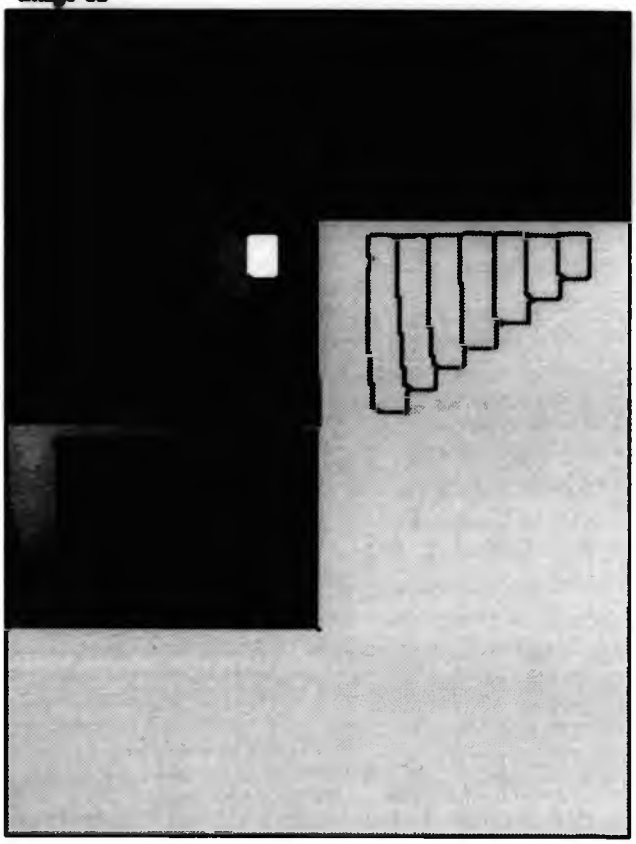
Image(6.7a) LUT of image(4.2b) & (4.2c) with 0.25 std-dev. gaussian noise.
Image(6.7b) LUT of image(4.2b) & (4.2c) with 0.50 std-dev. gaussian noise.
Image(6.7c) LUT of image(4.2b) & (4.2c) with 0.75 std-dev. gaussian noise.
Image(6.7d) LUT of image(4.2b) & (4.2c) with 1.00 std-dev. gaussian noise.
Image(6.7e) LUT of image(4.2b) & (4.2c) with 1.25 std-dev. gaussian noise.
Image(6.7f) LUT of image(4.2b) & (4.2c) with 1.50 std-dev. gaussian noise.
Image(6.7g) LUT of image(4.2b) & (4.2c) with 1.75 std-dev. gaussian noise.
Image(6.7h) LUT of image(4.2b) & (4.2c) with 2.00 std-dev. gaussian noise.
Image(6.7i) LUT of image(4.2b) & (4.2c) with 2.25 std-dev. gaussian noise.
Image(6.7j) LUT of image(4.2b) & (4.2c) with 2.50 std-dev. gaussian noise.
Image(6.7k) LUT of image(4.2b) & (4.2c) with 2.75 std-dev. gaussian noise.
Image(6.7l) LUT of image(4.2b) & (4.2c) with 3.00 std-dev. gaussian noise.
Image(6.7m) LUT of image(4.2b) & (4.2c) with 3.25 std-dev. gaussian noise.
Image(6.7n) LUT of image(4.2b) & (4.2c) with 3.50 std-dev. gaussian noise.
Image(6.7o) LUT of image(4.2b) & (4.2c) with 3.75 std-dev. gaussian noise.
Image(6.7p) LUT of image(4.2b) & (4.2c) with 4.00 std-dev. gaussian noise.
Image(6.7q) LUT of image(4.2b) & (4.2c) with 4.25 std-dev. gaussian noise.
Image(6.7r) LUT of image(4.2b) & (4.2c) with 4.50 std-dev. gaussian noise.
Image(6.7s) LUT of image(4.2b) & (4.2c) with 4.75 std-dev. gaussian noise.
Image(6.7t) LUT of image(4.2b) & (4.2c) with 5.00 std-dev. gaussian noise.
Image(6.7u) LUT of image(4.2b) & (4.2c) with 5.25 std-dev. gaussian noise.
Image(6.7v) LUT of image(4.2b) & (4.2c) with 5.50 std-dev. gaussian noise.
Image(6.7w) LUT of image(4.2b) & (4.2c) with 5.75 std-dev. gaussian noise.
Image(6.7x) LUT of image(4.2b) & (4.2c) with 6.00 std-dev. gaussian noise.

Image(6.8a) The segmentation using image(6.7a).
Image(6.8b) The segmentation using image(6.7b).
Image(6.8c) The segmentation using image(6.7c).
Image(6.8d) The segmentation using image(6.7d).
Image(6.8e) The segmentation using image(6.7e).
Image(6.8f) The segmentation using image(6.7f).
Image(6.8g) The segmentation using image(6.7g).
Image(6.8h) The segmentation using image(6.7h).
Image(6.8i) The segmentation using image(6.7i).
Image(6.8j) The segmentation using image(6.7j).
Image(6.8k) The segmentation using image(6.7k).
Image(6.8l) The segmentation using image(6.7l).
Image(6.8m) The segmentation using image(6.7m).
Image(6.8n) The segmentation using image(6.7n).
Image(6.8o) The segmentation using image(6.7o).
Image(6.8p) The segmentation using image(6.7p).
Image(6.8q) The segmentation using image(6.7q).
Image(6.8r) The segmentation using image(6.7r).
Image(6.8s) The segmentation using image(6.7s).
Image(6.8t) The segmentation using image(6.7t).
Image(6.8u) The segmentation using image(6.7u).
Image(6.8v) The segmentation using image(6.7v).
Image(6.8w) The segmentation using image(6.7w).
Image(6.8x) The segmentation using image(6.7x).

Image(6.9a) The image(4.2b) added with 1 std-dev. Gaussian noise.
Image(6.9b) The image(4.2c) added with 1 std-dev. Gaussian noise.
Image(6.9c) The image(4.2b) added with 2 std-dev. Gaussian noise.
Image(6.9d) The image(4.2c) added with 2 std-dev. Gaussian noise.
Image(6.9e) The image(4.2b) added with 3 std-dev. Gaussian noise.
Image(6.9f) The image(4.2c) added with 3 std-dev. Gaussian noise.
Image(6.9g) The image(4.2b) added with 4 std-dev. Gaussian noise.
Image(6.9h) The image(4.2c) added with 4 std-dev. Gaussian noise.
Image(6.9i) The image(4.2b) added with 5 std-dev. Gaussian noise.
Image(6.9j) The image(4.2c) added with 5 std-dev. Gaussian noise.
Image(6.9k) The image(4.2b) added with 6 std-dev. Gaussian noise.
Image(6.9l) The image(4.2c) added with 6 std-dev. Gaussian noise.

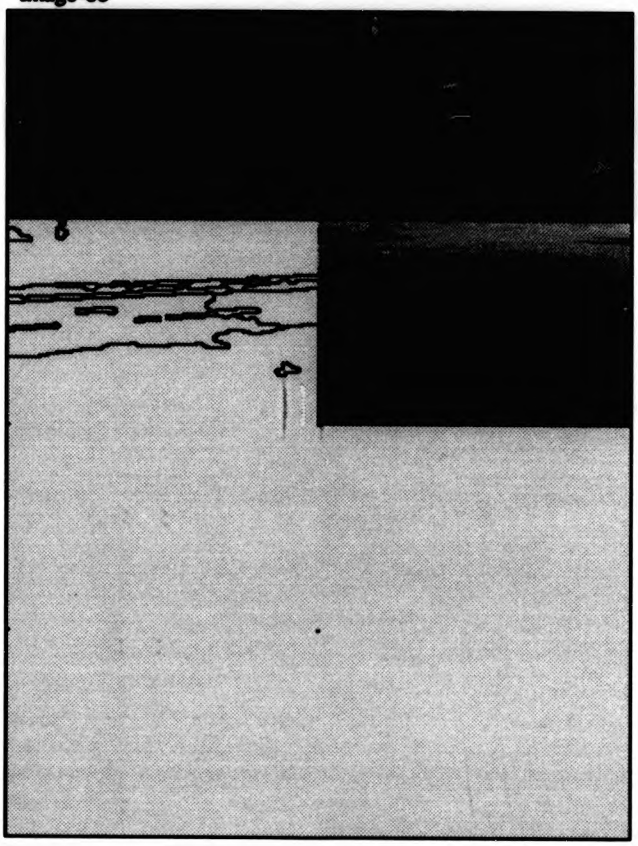
6.1a	6.1b
6.1c	6.1d
6.1e	

image-6a



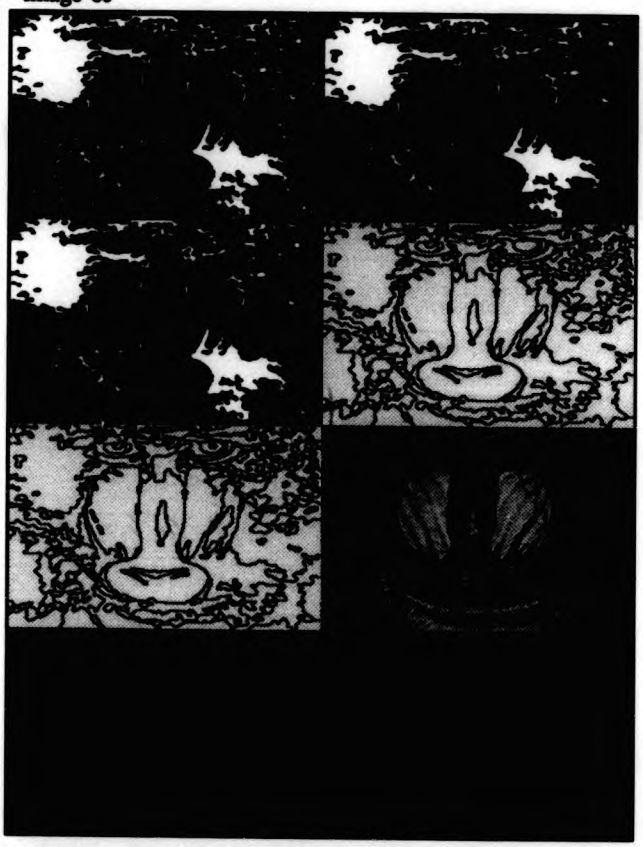
6.2a	6.2b
6.2c	6.2d

image-6b



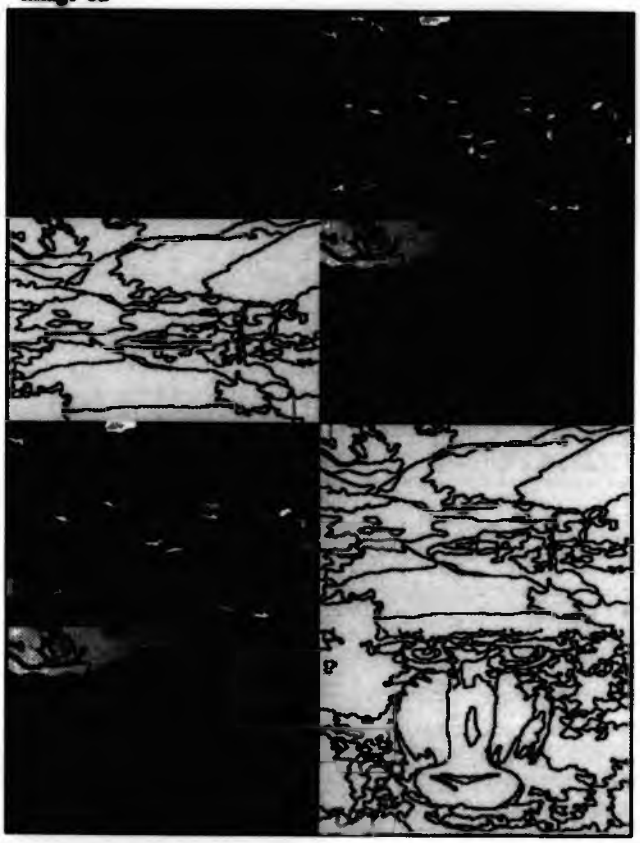
6.3a	6.3b
6.3c	6.3d
6.3e	6.3f
6.3g	6.3h

image-6c



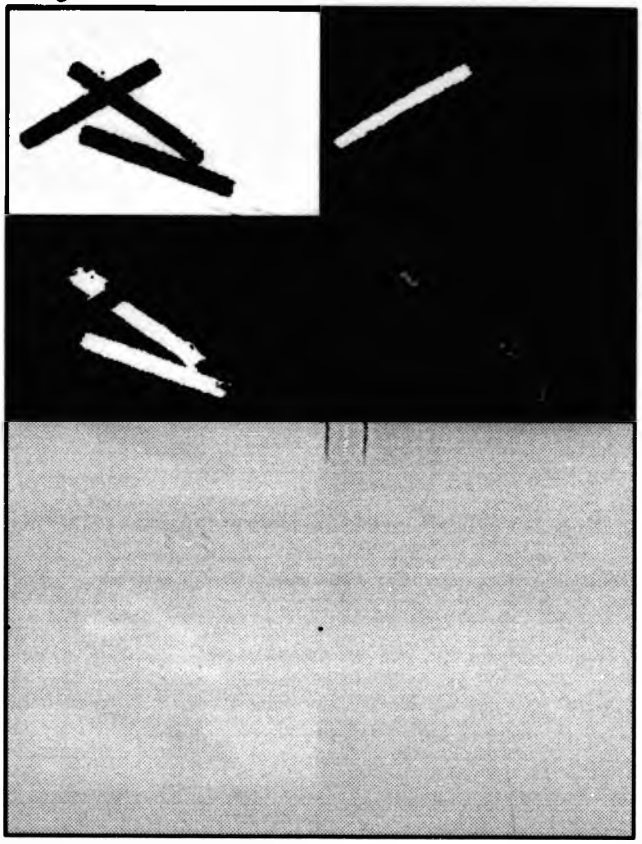
6.3i	6.4a
6.4b	6.4c
6.4d	6.4e
6.4f	6.5a

image-6d



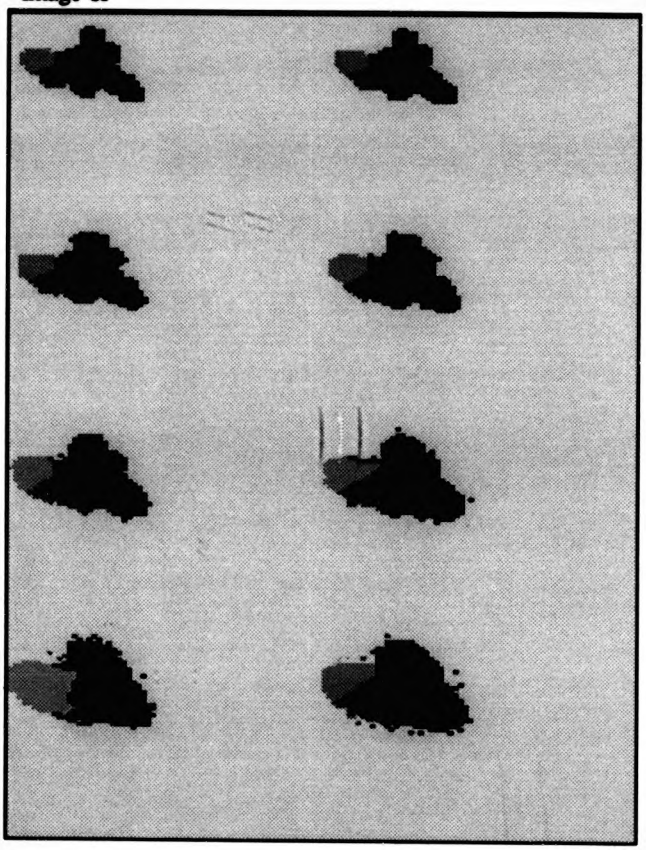
6.6a	6.6b
6.6c	6.6d

image-6e



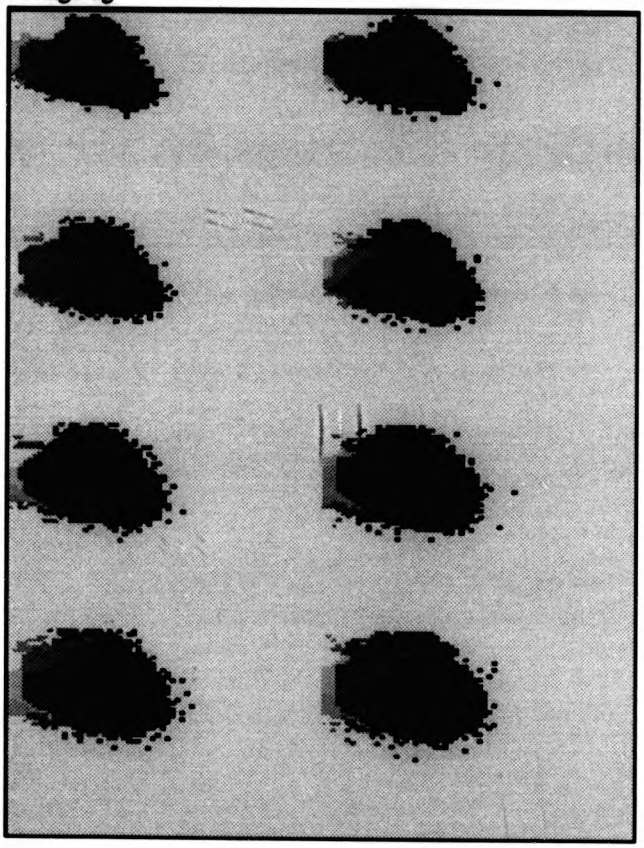
6.7a	6.7b
6.7c	6.7d
6.7e	6.7f
6.7g	6.7h

image-6f



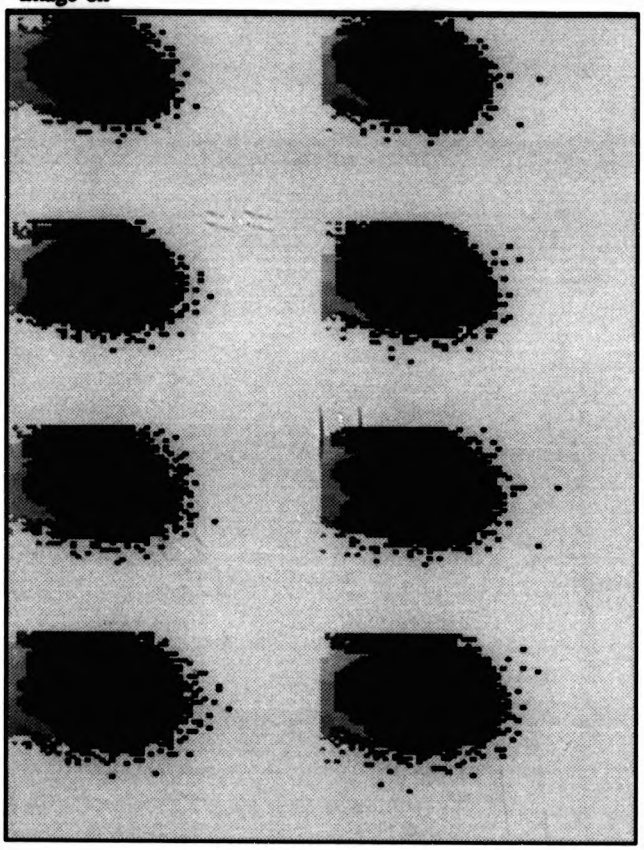
6.7i	6.7j
6.7k	6.7l
6.7m	6.7n
6.7o	6.7p

image-6g



6.7q	6.7r
6.7s	6.7t
6.7u	6.7v
6.7w	6.7x

image-6h

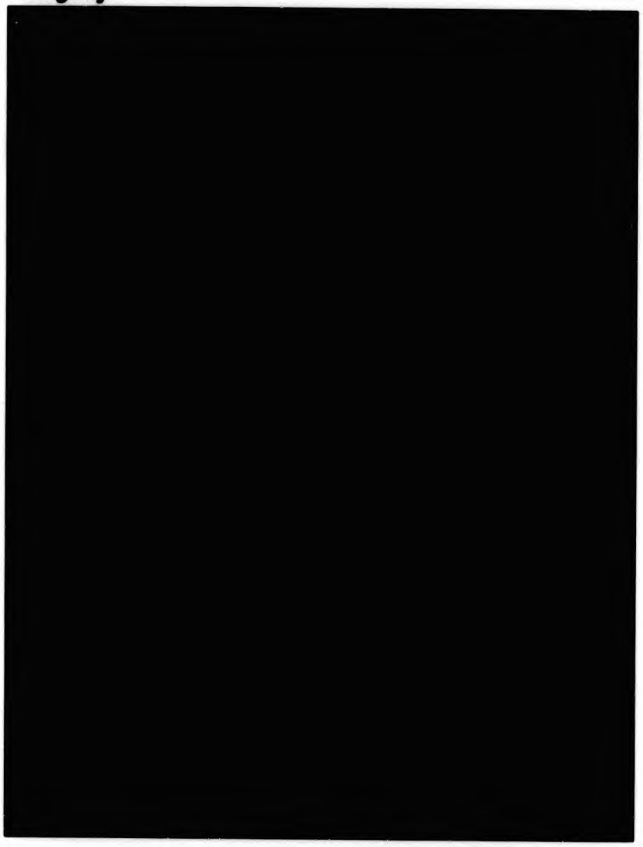


6.8a	6.8b
6.8c	6.8d
6.8e	6.8f
6.8g	6.8h

image-6i

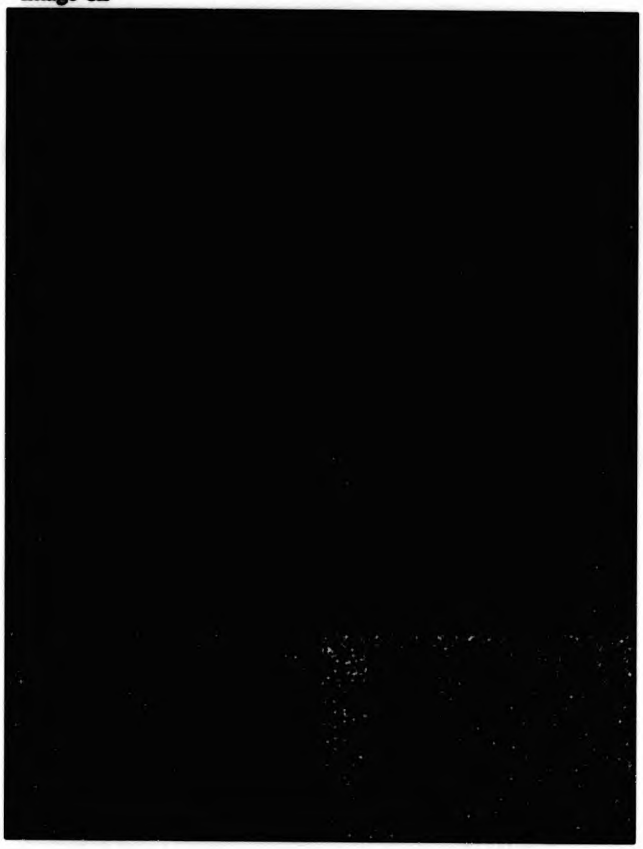
6.8i	6.8j
6.8k	6.8l
6.8m	6.8n
6.8o	6.8p

image-6j



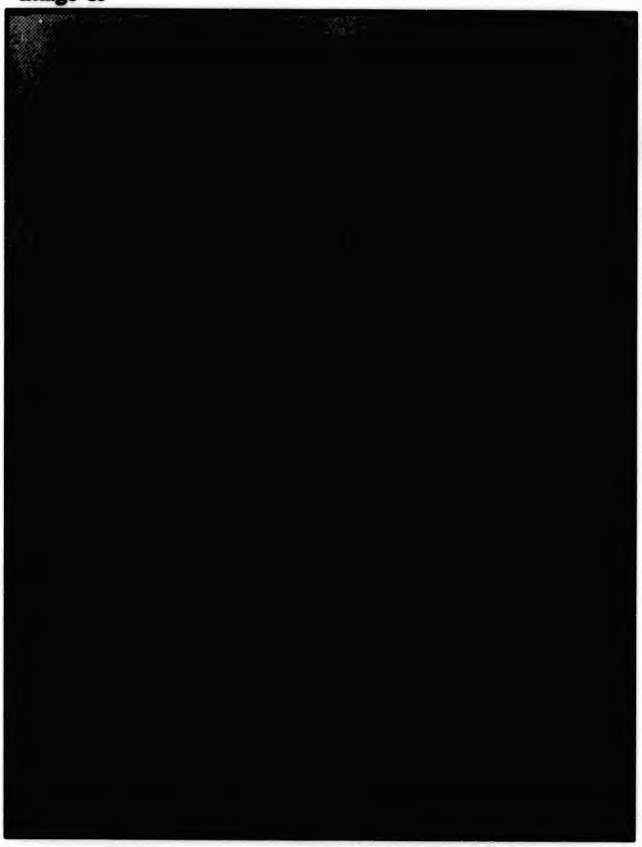
6.8q	6.8r
6.8s	6.8t
6.8u	6.8v
6.8w	6.8x

image-6k



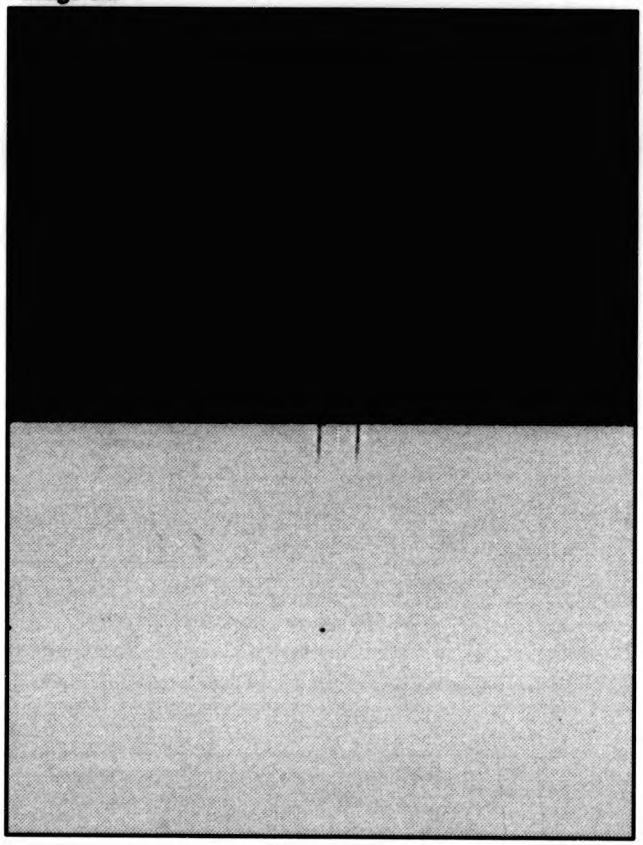
6.9a	6.9b
6.9c	6.9d
6.9e	6.9f
6.9g	6.9h

image-6l



6.9i	6.9j
6.9k	6.9l

image-6m



References

[Andreadis et al 90] I. Andreadis, M.A. Browne & J.A. Swift
Image pixel classification by chromaticity analysis
Pattern Recognition Letter 1990 Jan Vol.11(1) pp.51-58

[Asano et al 86] T. Asano, G. Kenwood, J. Mochizuki, & S. Hata
Color image recognition using Chrominance signals
Proc Of 8th Int Conf On Pattern Recognition, Pariisi 1986 pp.804-807

[Asano & Yokoya 81] T. Asano & N. Yokoya
Image segmentation scheme for low level computer vision
Pattern Recognition 1981 Vol.14 pp.267-273

[Aus et al 83] H.M. Aus, H. Harms, M. Haucke, B. Gerlach, A. Kriete & S. Thieme
Computer color vision
Proc 3rd Int Conf Robot Vision And Sensory Controls Cambridge MA USA 1983 pp.225-229

[Bajcsy 73] R. Bajcsy
Computer description of textured surfaces
Proc 3rd Int Joint Conf Artificial Intelligence 1973 pp.572-579

[Baker 80] H.H. Baker
Edge based stereo correlation
Proc ARPA Image Understanding Workshop University Of Maryland 1980 April pp.168-175

[Ballard & Brown 82] D.H. Ballard & C.M. Brown
Computer Vision
Prentice-Hall - (1982)

[Berger 71] T. Berger
Rate Distortion Theory : A Mathematical Basis for Data Compression
Englewood Cliffs, NJ: Prentice Hall - (1971)

[Berry 87] D.T. Berry
Colour recognition using spectral signatures
Pattern Recognition Letters 1987 Vol.6 pp.69-75

[Beucher 82] S. Beucher
Watersheds of function and picture segmentation
Proc IEEE Int Conf On Acoustics Speech And Signal Processing 1982 May pp.1928-1931

[Brice & Fennema 70] C.R. Brice & C.L. Fennema
Scene analysis using regions
Artificial Intell 1970 Vol.1 pp.205-226

[Broder & Rosenfeld 81] A. Broder & A. Rosenfeld
Gradient magnitude as an aid in color pixel classification
IEEE Trans System Man And Cybernatics 1981 Vol.11 pp.248-249

[Bruce & Green 90] V. Bruce & P.R. Green
Visual Perception Physiology, Psychology and Ecology
Lawrence Erlbaum Associates, Publishers - (1990)

[Burt et al 81] P.J. Burt, T.H. Hong, & A. Rosenfeld
Segmentation and estimation of region properties through co-operative hierarchical computation
IEEE Trans System Man And Cybernetics 1981 SMC-11 pp.802-809

[Castleman 79] K.R. Castleman
Digital Image Processing
Prentice-Hall - (1979)

[Celenk 90] M. Celenk
A colour clustering technique for image segmentation
Computer Vision, Graphics, and Image Processing 1990 Nove Vol.52(2) pp.145-170

[Chang et al 87] B.H. Chang, S.D. Kim & J.K. Kim
Threshold selection algorithm for extraction a uniform color region in an image
Electronics Letters 1987 Vol.23 No.25 pp.1362-1363

[Chassery & Garbay 83] J.M. Chassery & C. Garbay
Iterative process for color image segmentation using a convexity criterion
Proc Of The SPIE 1983 Vol.397 pp.165-172

[Chassery & Garbay 84] J.M. Chassery & C. Garbay
An iterative segmentation method based on contextual color and shape criterion
Proc 7th Int Jnt Conf On Pattern Recognition 1984 pp.642-644

[Chen & Milgram 82] M.J. Chen & D.L. Milgram
Binary color vision
Proc Of The 2nd Inter Conf On Robot Vision And Sensory Controls, Stuttgart, Germany, pp.293-306 November 2-4 (1982)

[Connah & Fishbourne 81] D. Connah & C. Fishbourne
The use of colour in industrial scene analysis
RoViSec 1 1981 pp.340-347

[Crisman & Thorpe 90] J.D. Crisman & C.E. Thorpe
Color vision for road following
in Vision and Navigation: the Carnegie Mellon Navlab, ed C.E. Thorpe, Kluwer Academic Publishers, 1990 pp.9-24

[Davis 75] L.S. Davis
A survey of edge detection techniques
Computer Graphics And Image Processing 1975 Vol.4 pp.248-270

[Ekstrom et al 84] M.P. Ekstrom et al
Digital Image Processing Techniques
Academic Press - (1984) pp.267-269

[Faugeras 79] O.D. Faugeras
Digital color image processing within the framework of a human visual model
IEEE Trans Acoust Speech Signal Process 1979 Vol.27 pp.380-393

[Freeman 74] H. Freeman
Computer processing of line drawing images
Computer Surveys 1974 March Vol.6 No.1 pp.57-98

[Fu et al 80] K.S. Fu et al
Digital Pattern Recognition
Springer-Verlag - 2nd edition (1980)

[Fu 82] K.S. Fu
Syntatic Pattern Recognition and Applications
Prentic-Hall - (1982)

[Fu & Mui 81] K.S. Fu & J.K. Mui
A survey of image segmentation
Pattern Recognition 1981 Vol.13 No.1 pp.3-16

[Funakubo 84] N. Funakubo
Region segmentation of biomedical tissue image using color texture feature
Proc 7th Int Conf On Pattern Recognition 1984 July pp.30-32

[Gallager 68] R.G. Gallager
Information Theory And Reliable Communication
Wiley, New York - (1968)

[Garbay 81] C. Garbay
Application of colored image processing to bone marrow cells recognition
Anal Quant Cytol And Histology 1981 Vol.4 No.3 pp.272-280

[Garbay 82] C. Garbay
A color metric as a tool for cytologic image analysis
Proc of the 1st ISMIII 1982 pp.311-315

[Giardina & Dougherty 88] C.R. Giardina & E.R. Dougherty
Morphological Methods in Image and Siginal Processing
Prentice Hall - (1988)

[Goldstein 89] E.B. Goldstein
Sensation and Perception
Wadsworth Publishing Company - (1989)

[Gonzalez 74] R.C. Gonzalez
Pattern Recognition Principles
Addison Wesley - (1974)

[Gonzalez & Wintz 87] R.C. Gonzalez & P. Wintz
Digital Image Processing
Addison Wesley - 2nd edition (1987)

[Gordillo 85] J.L. Gordillo
Colour representation for a vision machine
2nd Int Conference On Machine Intelligence 1985 pp.375-385

[Grimson 80] W.E.L. Grimson
A computer implementation of a theory of human stereo vision
Phil. Trans. Roy. Lond., B. 1980 Vol.292 pp.217-253

[Hall 79] E.L. Hall
Computer Image Processing and Recognition
Academic Press - (1979)

[Hara et al 83] Y. Hara, N. Akiyama & K. Karasaki
Automatic inspection system for printed circuit boards
IEEE Trans Pattern Anal Mach Intell 1983 Vol.5 No.6 pp.623-630

[Haralick 79] R.M. Haralick
Statistical and structural approaches to texture
Processings of the IEE 1979 Vol.67 No.5 pp.786-803

[Haralick et al 87] R.M. Haralick, S.R. Sternberg & X. Zhuang
Image analysis using mathematical morphology
IEEE Trans Pattern Anal Mach Intell 1987 Vol.9 pp.532-550

[Haralick & Dinstein 75] R.M. Haralick & I. Dinstein
A spatial clustering procedure for multi-image data
IEEE Trans Circuit And Systems 1975 Vol.22 pp.440-450

[Haralick & Shapiro 85] R.M. Haralick & L.G. Shapiro
Image segmentation techniques
Computer Vision Graphics and Image Processing 1985 Vol.29 No.1 pp.100-132

[Herault et al 89] J. Herault, C. Jutten & A. Guerin
Neural networks for data analysis in L2n
in Neural Network from Models to Applications, L. Personnaz G. Dreyfus, IDSET (1989)

[Hong & Rosenfeld 84] T.H. Hong, & A. Rosenfeld
Compact region extraction using weighted pixel linking in a pyramid
IEEE Trans Pattern Anal Mach Intell 1984 PAMI-6 pp.222-229

[Horowitz & Pavlidis 74] S.L. Horowitz & T. Pavlidis
Picture segmentation by a directed split-and-merge procedure
Proc 2nd Int Jnt Conf Pattern Recog 1974 pp.424-433

[Horowitz & Pavlidis 76] S.L. Horowitz & T. Pavlidis
Picture segmentation by a tree traversal algorithm
J. Ass. Comput. Mach. 1976 pp.368-388

[Hough 62] P.V.C. Hough
Methods and means for recognizing complex patterns
US Patent 3069654 1962

[Hueckel 71] M. Hueckel
An operator which locates edges in digitized pictures
Journal of the ACM 1971 Vol.18 pp.113-125

[Hueckel 73] M. Hueckel
A local visual operator which recognizes edges and lines
Journal of the ACM 1973 Vol.20 pp.634-647

[Huntsberger & Descalzi 85] T.L. Huntsberger & M.F. Descalzi
Color edge detection
Pattern Recognition Letter 1985 Vol.3 No.3 pp.205-209

[Ito 75] T. Ito
Color picture processing by computer
Fourth Joint Artificial Intelligence Conf 1975 pp.635-642

[Ito 76] T. Ito
Pattern classification by color effect method
Proc 3rd Int Jnt Conf On Pattern 1976 pp.26-30

[Ito 80] T. Ito
Digital color picture processing
Proc of EUSIPCO-80 1980 pp.71-81

[Ito & Fukushima 76] T. Ito & M. Fukushima
Computer analysis of color information with application to picture processing
Proc 3rd Int Jnt Conf On Pattern Recognition 1976 pp.833-837

[James 87] Mike James
Pattern Recognition
BSP Professional Book - (1987)

[Jordan III & Bovik 88] J.R. Jordan III & A.C. Bovik
Computataional stereo vision using color
IEEE Control System Magazine, 1988 Vol.8 No.3 pp.31-36

[Kak 84] A.C. Kak
Image reconstruction from projections
in Digital Image Processing Techniques, M.P. Ekstrom ed., Academic Press - (1984) pp.111-170

[Keil 83] R.E. Keil
Machine vision with color detection
Proc 3rd Int Conf Robot Vision And Sensory Controls, Cambrides, MA, USA. 1983 pp.503-512

[Keller et al 86] J.M. Keller, N. Covavisaruch, K. Unklesbay & N. Unklesbay
Color image analysis of food
IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami Beach, Florida 1986 June 22-26 pp.619-621

[Kelley & Faedo 85] R. Kelley & W. Faedo
A first look into color vision
SPIE Conf On Intell Robots And Computer Vision, Cambridge, MA, USA. 1985 pp.96-103

[Kender 77] J.R. Kender
Instabilities in color transformations
Conf. on Pattern Recognition and Image Processing, IEEE, Rensselaer Polytech. Institute, Troy, N.Y., June 1977 pp.266-274

[Khotanzad & Bouaría 90] A. Khotanzad & A. Bouaría
Image segmentation by a parallel, non-parametric histogram based clustering algorithm
Pattern Recognition 1990 Vol.23(9) pp.961-973

[Klinger & Dyer 76] A. Klinger & C.R. Dyer
Experiments on picture representation using regular decomposition
Computer Graphics And Image Processing 1976 Vol.5 pp.68-105

[Klinker et al 87] G.J. Klinker, S.A. Shafer & T. Kanade
Using a color reflection model to separate highlights from object color
Proceedings of the First International Conference on Computer Vision June 8-11 1987 pp.145-150

[Klinker et al 88] G.J. Klinker, S.A. Shafer & T. Kanade
The measurement of highlights in color images
International Journal of Computer Vision 1988 pp.7-32

[Klinker et al 90] G.J. Klinker, S.A. Shafer & T. Kanade
A physical approach to colour image understanding
Inter Journal of Computer Vision 1990 Jan Vol.4(1) pp.7-38

[Kohler 81] R. Kohler
A segmentation system based on thresholding
Computer Graphics And Image Processing 1981 Vol.15 pp.319-338

[Land 77] E.H. Land
The retinex theory of color vision
Sci AM 1977 Dec pp.108-128

[Lantuejoul 82] C. Lantuejoul
Geodesic segmentation
in Multicomputer And Image Processing, ed. K. Preston, Jr. and L. Uhr, Academic Press, 1982 pp.111-124

[Levine 78] M.D. Levine
A knowledge-based computer vision system
in Computer Vision Systems, A.R. Hanson and E.M. Riseman eds., New York : Academic Press - (1978)

[Levine 80] M.D. Levine
Region analysis using a pyramid data structure
in Structured Computer Vision, S. Tanimtoto and A. Klinger eds., Academic Press - (1980) pp.57-100

[Levine 85] M.D. Levine
Vision in Man and Machine
New York : McGraw-Hill - (1985)

[Maragos & Schafer 90] P. Maragos & R.W. Schafer
Morphological systems for multidimensional signal processing
Proc Of The IEEE 1990 Vol.78 No.4 pp.690-710

[Marr & Hildreth 80] D. Marr & E. Hildreth
Theory of edge detection
Proc. Royal Soc. 1980 Vol.207 pp.187-217

[Matheron 75] G. Matheron
Random Sets and Integral Geometry
J. Wiley & Sons Inc. - (1975)

[Milgram & Herman 80] D.L. Milgram & M. Herman
Clustering edge values for threshold selection
Computer Graphics And Image Processing 1980 Vol.10 pp.272-280

[Milgram & Kahl 79] D.L. Milgram & D.J. Kahl
Recursive region extraction
Computer Graphics And Image Processing 1979 Vol.9 pp.82-88

[Morris et al 86] O.J. Morris, M. de J. Lee & A.G. Constantinides
A unified method for segmentation and edge detection using graph theory
Proc ICASSP 86: IEEE Int Conf Acoust Speech & Sig Process., Tokyo 1986 pp.2051-2054

[Nazif & Levine 84] A.M. Nazif & M.D. Levine
Low level image segmentation: an expert system
IEEE Trans Pattern Anal Mach Intell 1984 Vol.6 pp.555-577

[Nevatia 76] R. Nevatia
A color edge detector
Proc 3rd Int Jnt Conference On Pattern Recognition 1976 pp.829-832

[Nevatia 77] R. Nevatia
A color edge detector and its use in scene segmentation
IEEE Trans System Man And Cybernetics 1977 Vol.7 pp.820-826

[Ohlander et al 78] R. Ohlander, K. Price & D.R. Reddy
Picture segmentation using a recursive region splitting method
Computer Graphics And Image Processing 1978 Vol.8 pp.313-333

[Ohta et al 80] Y. Ohta, T. Kanade & T. Saksi
Color information for region segmentation
Computer Graphics And Image Processing 1980 Vol.13 pp.222-241

[Ohta 85] Y. Ohta
Knowledge-based Interpretation of Outdoor Natural Color Scenes
Pitman Advanced Publishing Program - (1985)

[Parvin 84] B.A. Parvin
A split and merge algorithm for segmentation of natural scenes
Proc 7th Int Conf On Pattern Recognition 1984 pp.294-296

[Pavlidis & Liow 90] T. Pavlidis & Y.T. Liow
Integration region growing and edge detection
IEEE Trans Pattern Anal Mach Intell 1990 Vol.12 No.3 pp.225-233

[Pearson & Robinson 85] D.E. Pearson & J.A. Robinson
Visual communication at very low data rates
IEEE Proc 1985 Vol.72 pp.795-812

[Plummer 90] P. Plummer
The benefits of colour
Image Processing 1990 Sept/Oct pp.23-24

[Pratt 78] W.K. Pratt
Digital Image Processing
New York : Wiley-Interscience - (1978)

[Riseman & Arbib 77] E.M. Riseman & M.A. Arbib
Computational techniques in the visual segmentation of static scenes
Computer Graphics And Image Processing 1977 Vol.6 pp.221-276

[Roberts 64] L.G. Roberts
Machine perception of three dimensional solids
Symposium of Optical and Electro-Optical Information Processing Technology, Boston MA, pp159-197 (1964)

[Robinson 76] G.S. Robinson
Color edge detection
Proc SPIE Symp Advances In Image Transmission Techniques 1976 Vol.87 pp.126-133

[Rosenfeld & Kak 82] A. Rosenfeld & A.C. Kak
Digital Picture Processing
New York : Academic Press - 2nd edition (1982), Volume 1 & 2.

[Rosenfeld & Thurston 71] A. Rosenfeld & M. Thurston
Edge and curve detection for visual scene analysis
IEEE Trans On Computer 1971 Vol.20 pp.562-569

[Rosenfeld & Troy 70] A. Rosenfeld & Eleanor Troy
Visual texture analysis
Conf Record For Symposium On Feature Extraction And Selection In Pattern Recognition (IEEE Publication 70C-51C), Argonne, Illinois, Oct 1970 pp.115-124

[Boynton 79] R.M. Boynton
Human Color Vision
Holt, Rinehart and Winston - (1979)

[Sarabi & Aggarwal 81] A. Sarabi & J.K. Aggarwal
Segmentation of chromatic images
Pattern Recognition 1981 Vol.13 pp.417-427

[Schacter et al 76] B.J. Schacter, L.S. Davis & A. Rosenfeld
Scene segmentation by cluster detection in color spaces
SIGART Newsletter 1976 Vol.58 pp.16-17

[Serra 80] J. Serra
Image Analysis And Mathematical Morphology
Academic Press - (1980)

[Serra 88] J. Serra
Image Analysis And Mathematical Morphology - Vol.2
Academic Press - (1988)

[Shafer 80] S.A. Shafer
MOOSE: User's Manual, Implementation Guide, Evaluation
IFI-HH B-70/80, Fachbereich Informatik, University of Hamburg, Hamburg, West Germany, April (1980)

[Shafer & Kanade 82] S. Shafer & T. Kanade
Recursive region segmentation by analysis of histogram
Int Conf On Acoustic Speech And Signal Processing, 1982 May pp.1166-1171

[Shannon & Weaver 49] C.E. Shannon & W. Weaver
The mathematical theory of communication
Bell Syst Tech J 1949 Vol.27 pp.379-423,623,656

[Shirai 87] Y. Shirai
Three-Dimensional Computer Vision
Springer-Verlag - (1987)

[Sobel 70] I. Sobel
Camera models and machine perception
Stanford Artificial Intelligence Projects, AIM-121, Dept. of Computer Science, Stanford University, Stanford CA, (1970)

[Sternberg 79] S.R. Sternberg
Parallel architectures in image processing
Proc. IEEE Conf. Comput. Softw. Applic., Chicago 1979 pp.712-717

[Sternberg 83] S.R. Sternberg
Biomedical Image Processing
IEEE Computer 1983 Vol.16 pp.22-34

[Suk & Cho 83] M. Suk & T.H. Cho
Segmentation of image using minimum spanning trees
Proc SPIE 1983 Vol.397 pp.180-185

[Tanimoto & Pavlidis 75] S. Tanimoto & T. Pavlidis
A Hierarchical data structure for picture processing
Computer Graphics And Image Processing 1975 Vol.4 pp.104-119

[Thomas & Connolly 86] W.V. Thomas & C. Connolly
Applications of color processing in optical inspection
Proc Of The Society Of Photo-optical Instrumentation Engineer 654 (Automatic Optical Inspection) 1986 pp.116-122

[Tominaga 83] S. Tominaga
Computer perception of color measured with an image input device
Proc. IEEE Conf. on CVPR, Washington 1983 June pp.225-230

[Tominaga 84] S. Tominaga
A mapping method for computer color vision
Proc 7th Int Jnt Conf On Pattern Recognition 1984 pp.650-652

[Tominaga 86] S. Tominaga
Color image segmentation using three perceptual attributes
IEEE Conf On Computer Vision And Pattern Recognition 1986 June pp.628-630

[Tominaga 87] S. Tominaga
Expansion of color images using three perceptual attributes
Pattern Recognition Letters 1987 Vol.6 pp.77-85

[Tominaga 88] S. Tominaga
A color classification algorithm for color images
Pattern Recognition 4th Inter Conference Cambridge, U.K., March 1988 Proceedings pp.163-172 Springer-Verlag

[Tomita et al 73] F. Tomita, M. Yachida, & S. Tsuji
Detection of homogeneous regions by structural analysis
Proce Of The Third Inter Joint Conf On Arti Intell 1973 pp.564-571

[Tomita & Tsjui 90] F. Tomita, & S. Tsuji
Computer Analysis of Visual Texture
Kluwer Academic Publisher - (1990)

[Underwood & Aggarwal 77] S.A. Underwood & J.K. Aggarwal
Interactive computer analysis of aerial colour infrared photographs
Computer Graphics And Image Processing 1977 Vol.6 pp.1-24

[Wald & Brown 65] G. Wald & P.K. Brown
Human color vision and color blindness
Cold Spring Harbor Symposia on Quantitative Biology 1965 pp.345-359

[Watson 87] A.I. Watson
A new method of classification for landsat data using the watershed algorithm
Pattern Recognition Letters 1987 Vol.6 No.1 pp.15-19

[Weszka 78] J.S. Weszka
A survey of threshold selection techniques
Computer Graphics and Image Processing 1978 pp.259-265

[Wyszecki & Stiles 82] G. Wyszecki & W.S. Stiles
Color Science
John Wiley & Sons - 2nd edition (1982)

[Yasnoff et al 77] W.A. Yasnoff, J.K. Mui & J.W. Bacus
Error measure for scene segmentation
Pattern Recognition 1977 Vol.9 pp.217-231

[Zucker 76] S.W. Zucker
Region Growing: childhood and adolescene
Computer Graphics And Image Processing 1976 Vol.5 pp.382-399