Chapter 1

Introduction

Image segmentation is a technique to obtain a compact, summary representation of an image [1]. This representation is meaningful and often can be used to improve the performance of the followed image processing, if possible. For example, if we want to find an object in the image, searching each pixel in the image to find out the object is not very efficient. Instead, if the image could be represented by some regions that possibly comprise the object in one of them, then the object searching can take advantage of this representation to find out the object more efficiently or accurately. Of course, the representation may depend on the problem we are going to deal with. If we are browsing a large collection of video sequences, they may be represented by subsequences that look similar; if we are searching people in an image, body segments might be first identified, and then be assembled to form a people. Despite of the dependency on the problems, image segmentation still often plays an important role as a preprocessing to improve the performance of many applications, such as object searching [2] [3], pattern recognition [4] and image or video compression [5].

It is not easy to judge what segmentation of an image is good. In fact, good segmentation can be defined to be segmentation that well matches our expectation for

a specific problem. For instance, if we are doing license plate recognition, it is expected that the license plate is just enclosed in one of the regions represented by the image segmentation. However, the best segmentation is usually not available because of the diversity of the input images. Therefore, except for good segmentation, the results of image segmentation can be classified into over-segmentation and under-segmentation [6]. Segmentation is said to be over-segmentation if the object interesting is excessively divided into many parts; and it is said to be under-segmentation if the object interesting is enclosed in one of the segmented regions. For over-segmentation, it might be necessary to assemble several segmented regions to recover the object interesting, and the assembled regions might worse become under-segmentation. Besides, assembling segmented regions to recover the object interesting may be difficult because the characteristics of the object have been divided into several fractions. As a result, under-segmentation is obviously more preferable than over-segmentation because under-segmentation retains the integrity of the object interesting and can provide useful information to facilitate the following image processing.

In chapter 2, some conventional image segmentation methods and their corresponding problems are briefly reviewed. Chapter 3 and 4 respectively describe the first and second stages of the image segmentation method in this thesis. Some

experimental results and comparison are given in chapter 5. And this thesis ends up with a conclusion given in chapter 6.

Chapter 2

Conventional Image Segmentation Methods and Their Problems

It is known that image segmentation methods can be mainly classified into two categories, contour-based and region-based approaches [7]. These two types of image segmentation approaches and their inherent problems are respectively reviewed here briefly to give an overview for problems in image segmentation.

2.1 Contour-Based Methods

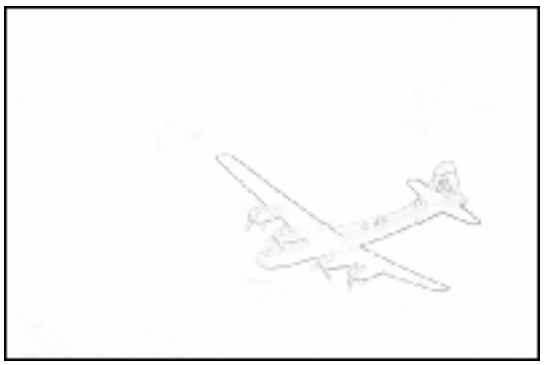
As its name indicates, contour-based methods take advantage of the contour information in the image to accomplish image segmentation. In general, contour analysis can be done by a two-stages processing. In the first stage, an edge detection technology, such as Sobel [8], Canny [9], or Oriented Energy edge detectors [10], is usually applied to the image. After edge detection, an image which reveals the probability of pixels being edge can be obtained, i.e., pixels with large brightness change in the original image will have high response in edge detection. Usually, this image is further binarized by a threshold. The binarization treats the pixels with values greater than the threshold as edge pixels and others as non-edge pixels. These edge pixels are usually assembled to form more meaningful edges by linking

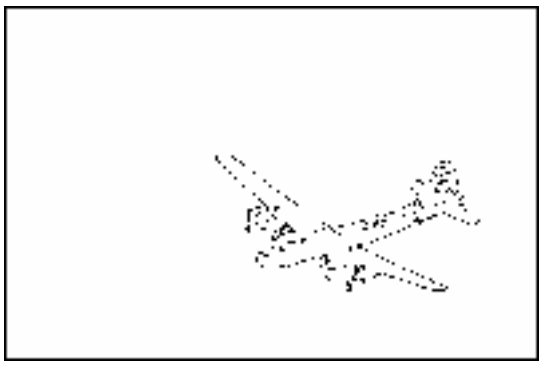
procedures in the second stage.

Consider an untextured image shown in fig. 2.1(a). The result of the image processed by the Sobel edge detector is shown in fig. 2.1(b), and the binarized image of fig. 2.1(b) with a threshold 10 is shown in fig 2.1(c). The binarization is to treat pixels with values greater than the threshold as edge pixels. The edge pixels in 2.1(c) reasonably correspond to the object boundary, and, after edge linking, can easily achieve the actual object boundary. Therefore, for untextured images, satisfactory results of segmentation usually can be obtained by contour-based methods. However, in the case of textured images, in addition to the pixels corresponding to the boundary of the objects, the pixels related to the grain in the texture may be also detected in the edge detection process. Hence, further using edge linking process to these detected pixels will yield wrong results, i.e., the boundary of the objects can not be appropriately identified. To illustrate this, consider the results of edge detection for the image in fig. 2.2(a), shown in fig. 2.2(c) and (d) which are binarized from (b) with thresholds 0.2 and 0.1, respectively. It is very difficult to set a threshold for the edge detector to obtain a group of edge pixels, which could correctly indicate the location of the object boundary in the image. If the threshold value is set high, boundary could be better identified in the texture area, but corrupted poor detection for the real object boundary could also be obtained. On the other hand, if the threshold value is set low,

in addition to real object boundaries, the grain in the texture would be also treated as object boundaries. Thus, both threshold values, too high and too low, would produce wrong object boundary detection or, moreover, unnecessarily complicate the processing of the following edge linking, i.e., it is very difficult to set an adequate threshold value, which can suppress wrong edge pixels corresponding to the grain inside the texture and let pass the edge pixels corresponding to real object boundary. As a result, the contour-based method is useful to determine the boundary of non-textured regions in an image while not suitable for textured regions.

(a)





(b) (c)

Figure 2.1 (a) An untextured image, (b) a result using Sobel operator, and (c) a binarized image

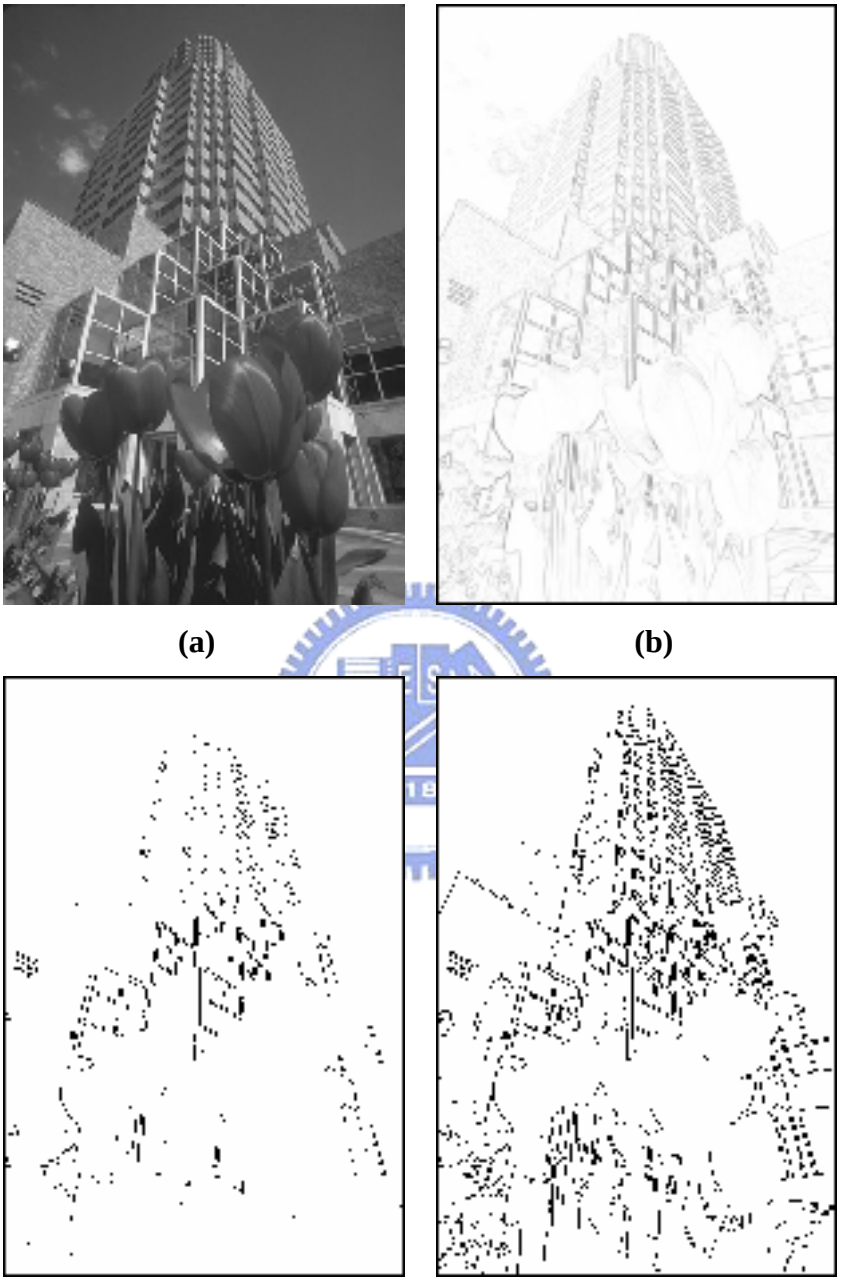


Figure 2.2 (a) A textured image, (b) a result using Sobel operator, and binarized images using threshold (c) 0.2 and (d) 0.1

2.2 Region-Based Methods

The most commonly used region-based segmentation methods include region growing, and region merging and splitting [8]. Typical region-based method first chooses one or more starting points in the image, called seeds, and then appending to these seeds their adjacent pixels recursively with a merging criterion. The generation of seed points may depend on the problem under consideration. However, this information is usually not available if the image data is not limited to a specific kind, such as images that contain some known objects. Generally, features of each pixels corresponding to some properties are computed, and if they form some clusters in the feature space, pixels corresponding to the center of these clusters will be treated as seeds. On the other hand, the stopping rule for region growing is also a problem. In general, the growing process will stop when no more pixels satisfy the merging criterion. If the images to be processed are limited to a specific category, then expected results can be obtained more effectively by the use of additional information such as color, texture, size, and shape.

The above discussion reveals that if the content of the input images have some special property, then both seed generation and the stopping rule for region growing can be more effectively determined. Therefore, region-based methods are more powerful for specific kind of images.

2.3 Motivation of the Hierarchical Segmentation Method

Conventional contour-based image segmentation methods are more suitable for non-textured images and region-based image segmentation methods can be more powerful for images with special properties. Therefore, the necessary of a method is evident to find the boundaries of objects in an image consisting of both textured and non-textured regions. Commonly, the object boundaries in an image can be found from many cues simultaneously, including the edges corresponding to the brightness, color, and textures [4], [7], [11]. In addition to these frequently used cues, other properties such as proximity, similarity, common region, and familiar configuration can also help in image segmentation [1]. Although these cues may facilitate the performance of image segmentation, it is not easy to consider all of them at the same time, and inclusion of a new cue may result in a reconstruction of a new segmentation method. Therefore, a hierarchical segmentation which deals with one cue in each stage of segmentation is adopted. Different to the method proposed in [12], the method in this thesis provides a segmentation scheme in a coarse-to-fine manner. The hierarchical processing has an advantage of separate the consideration of cues and hence could simplify the image segmentation problem. In addition, this method could easily integrate some other information, such as geometry and range data, for improvement of the performance of segmentation, if possible.

Despite of the dependency for image segmentation on problems, this thesis tries to provide a general segmentation method which may be direct used or easily modified to tackle the problems on hand. Based on the assumption that, for natural images, an object is defined as a group of connected pixels with similar color and texture, the goal of the image segmentation method in this thesis tries to under-segment the image into regions respectively with coherent color and texture.

This image segmentation method proposed a two stages segmentation related to color and textures, respectively. Fig. 2.3 shows the processing flow of this segmentation method. The original image is first processed by a Gaussian low pass filter to reduce the influence of noise and then contiguously processed by color and texture segmentation to accomplish the whole segmentation.

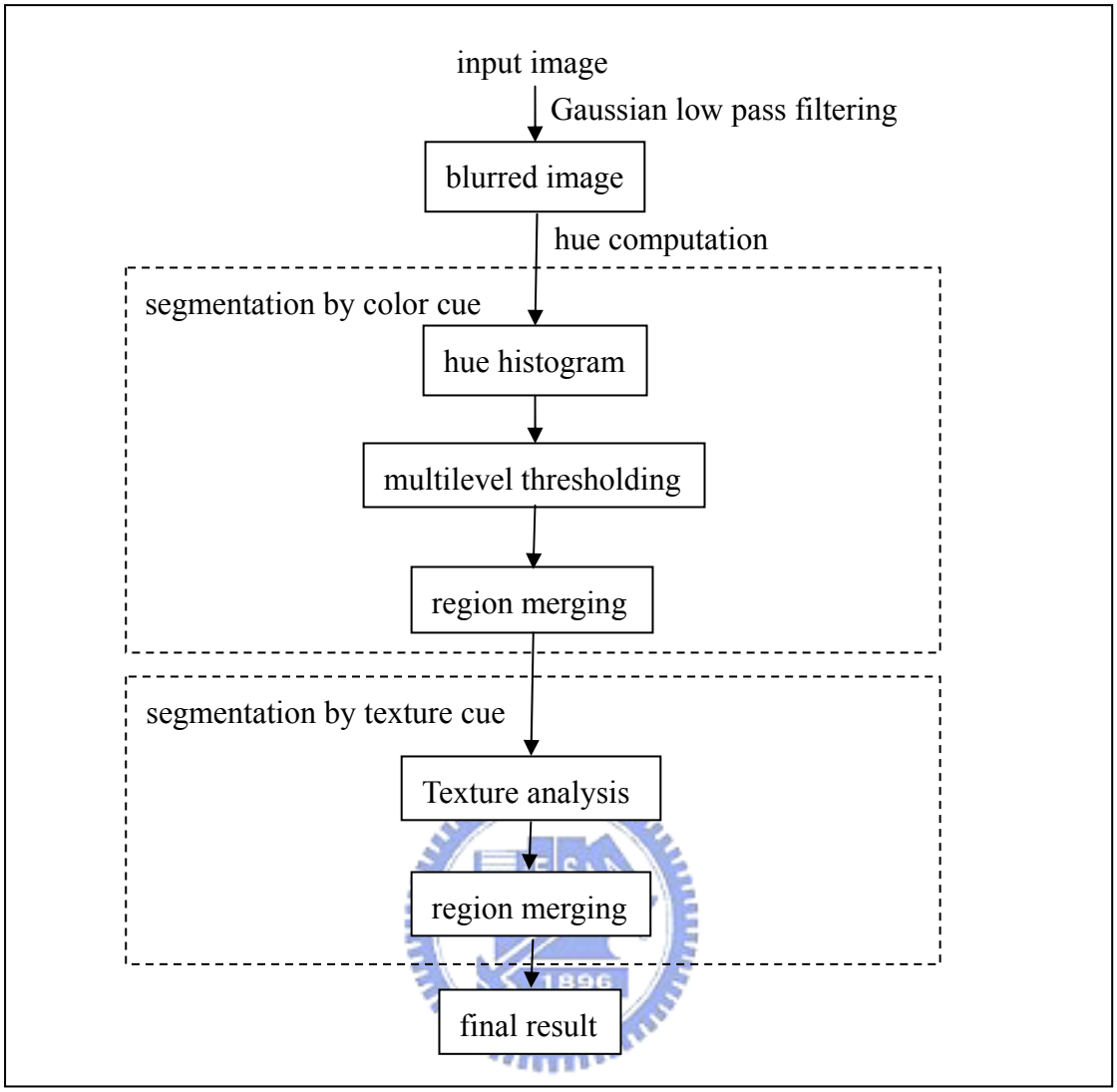
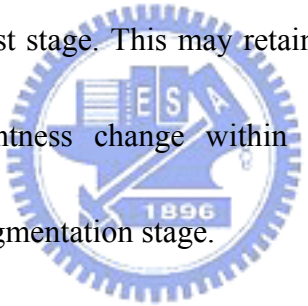


Figure 2.3 The processing flow of the image segmentation method in this thesis

Chapter 3

Image Segmentation by Color Cue

It is usually expected that an object in a natural image would possess similar color with some variable brightness caused by the texture in the object or non-uniform illumination. Due to the fact that the eye has less acuity for color than for luminance [8] and color feature hue has resistance to the non-uniform illumination caused by shade, shadow, reflect lights, and so on [13], the color feature is exploited to obtain a coarse segmentation in the first stage. This may retain the integrity of objects, avoid the disturbance of the brightness change within an object, and allow further processing in the following segmentation stage.

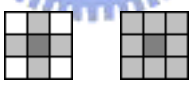


3.1 Some Relationships between Pixels

To continue our discussion, some spatial relations between pixels are reviewed in this section. 4-adjacency and 8-adjacency, the two most commonly used types of adjacency, are introduced here. Assume pixel p is at coordinate (x,y) and pixel q is at (s,t). Two pixels p and q are said to be 4-adjacency if pixel q is at $(x+1,y)$, $(x-1,y)$, $(x,y+1)$, or $(x,y-1)$, and vice versa. Two pixels p and q are said to be 8-adjacency if pixel q is at $(x+1,y)$, $(x-1,y)$, $(x,y+1)$, $(x,y-1)$, $(x+1,y+1)$, $(x-1,y+1)$, $(x+1,y-1)$, or

(x-1,y-1), and vice versa. These two types of adjacency are illustrated in fig 3.1. The light gray pixels are said to be 4- and 8- adjacency with the central points in fig. 3.1(a) and (b), respectively.

Let S be a subset of pixels in an image. Two pixels are said to be connected in S if, with 4- or 8-adjacency, there is a path between them consisting entirely of the other pixels in the set. For any pixel in S, the set of pixels that are connected to it in S is called a connected component of S. If a set of pixels has only one connected component, then the set is called a connected set or a region [8]. Fig. 3.2 shows a set of points to be processed (dark gray). There are 5 regions if they are viewed with 4-adjacency and 4 regions with 8-adjacency.



(a) (b)

Figure 3.1 Images to demonstrate (a) 4-adjacency, and (b) 8-adjacency

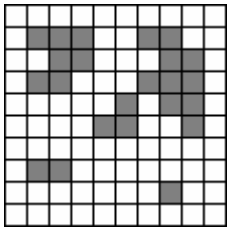


Figure 3.2 An example to demonstrate connected components

3.2 Transformation to HSI Color Space

Color images are most commonly stored and described by the combination of three color primaries, Red, Green, and Blue. Images described in such way is said to be presented in RGB color space. RGB color space makes sense because human eye has strong perception to these three color primaries, and hardware implementation of RGB color space is suited and straightforward. Fig. 3.3 shows an RGB color cube to illustrate the RGB color space.

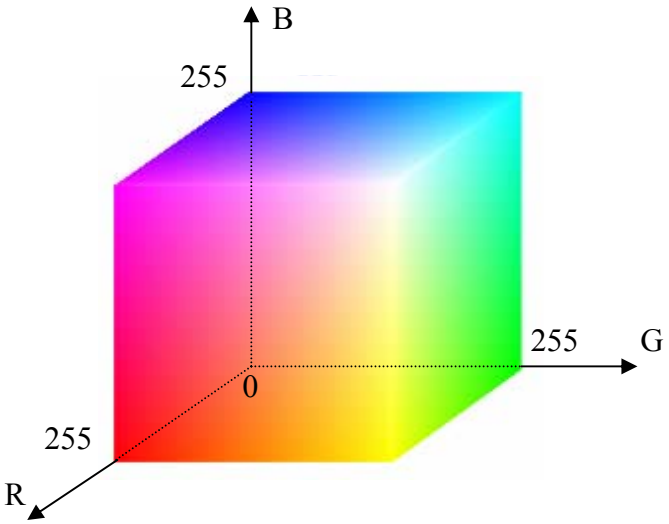


Figure 3.3 RGB color cube

Although RGB color space is very useful, it does not intuitively match human interpretation of color, i.e., the use of RGB percentage is not the way fulfilled in human eyes. Rather than in the RGB color space, a color to humans is more intuitively described with hue, saturation, and intensity in the HSI color model, which is obtained by transforming the RGB color space to a cylindrical space. The

coordinate along the cylindrical axis represents the brightness, which corresponds to the points of R=G=B in the RBG color space. The coordinate to represent saturation is along the radius perpendicular to the brightness axis and measures the distance to the brightness axis. Finally, the coordinate to represent hue is the angle between the radius axis and a reference direction corresponding to red color. The transformation formulas from RGB to HSI are

$$H = \begin{cases} \theta & B \le G \\ 360 - \theta & B > G \end{cases} \tag{3-1}$$

$$S = 1 - \frac{3[\min(R,G,B)]}{(R+G+B)} \tag{3-2}$$

$$I = \frac{(R+G+B)}{3} \tag{3-3}$$

with

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2 + (R-B)(G-B)]^{1/2}} \right\} \tag{3-4}$$

The hue stands for the name of the color. For example, 0 degree means red, 120 degree means green, 240 degree means blue, and 60 degree means yellow. The saturation indicates how much the color is mixed by white color. The intensity concerns about the brightness intensity. Note that the range of θ is from 0 to 359 degree, S is from 0 to 1, and I is from 0 to 255 if R, G, and B are respectively represented by 8 bits and has values from 0 to 255. Moreover, hue is not defined in the case that R=G=B. Fig. 3.4 shows a RGB color cube fitted by the HSI coordinate,

and an intersection of the RGB color cube and a plane for certain I value to illustrate the HSI color model.

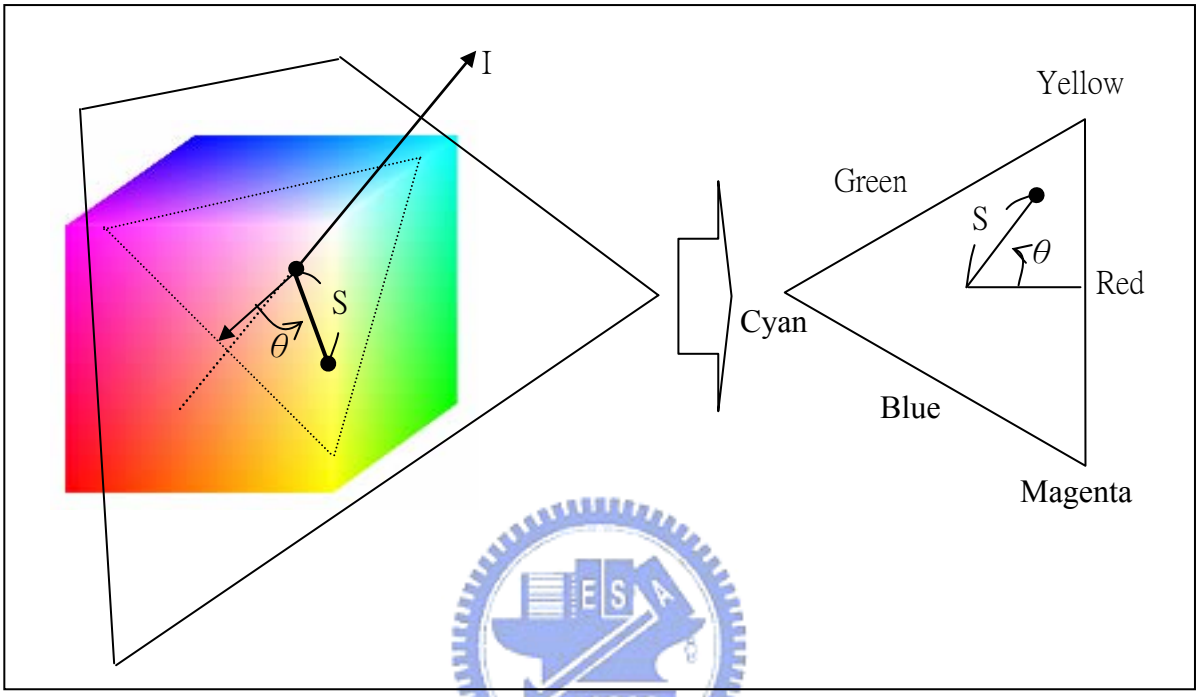


Figure 3.4 HSI color model

For pixels with R=G=B, the denominator within the arc cosine function in equation (3-4) will be zero and thus the hue for these pixels is not defined. In other words, although hue is useful as an index of color, it is not defined for gray pixels because of their lack of color information. Usually, there are hue undefined pixels after RGB to HSI transformation and, for natural images, the hue undefined points often from small regions scattered over the image. The hue of these pixels can be computed by the average value of the hue of the adjacent pixels. This makes sense

because in natural images, hue undefined small region often appears due to noise and quantization error during image acquisition and storage.

When only single hue undefined point appears, it is probably a noisy point because its adjacent points all possess their own hue values, respectively. Therefore, this hue undefined point can be easily removed by the average value of the hue of the 8 neighbors. When a region of hue undefined points appears, the following procedure is used to eliminate this region:

(1) Find out the points which have most neighbors that have hue.

(2) Determine the hue for these points respectively by the average of hue of the neighbors that have hue.

(3) Treat the remaining hue undefined region in step 2 as a new one and repeat this procedure until all hue undefined points are eliminated.

Fig. 3.5 is an example to demonstrate the method of eliminating the hue undefined region. Fig. 3.5(a) shows a hue undefined region, labeled by white and enclosed by points with hue calculated by equation (3-1), which are labeled by dark gray. This hue undefined region only has one point with 7 neighbors whose hue had been calculated, and it is labeled as light gray in (b). Thus in step (2), only the hue of the light gray point in (b) has to be determined. This is done by averaging the hue of the 7 dark gray neighbors. The average value is filled into the light gray point and, as

a result, a new hue undefined region is produced as in (c). Repeat the procedure for (c). To find points with most hue calculated neighbors, only 3 points with 6 hue calculated neighbors are found as the light gray in (d). Then, the hue of these 3 points are respectively filled in by the average of hue of its 6 hue calculated neighbors. Fig. 3.5 (e) to (n) contiguously finished the whole process.

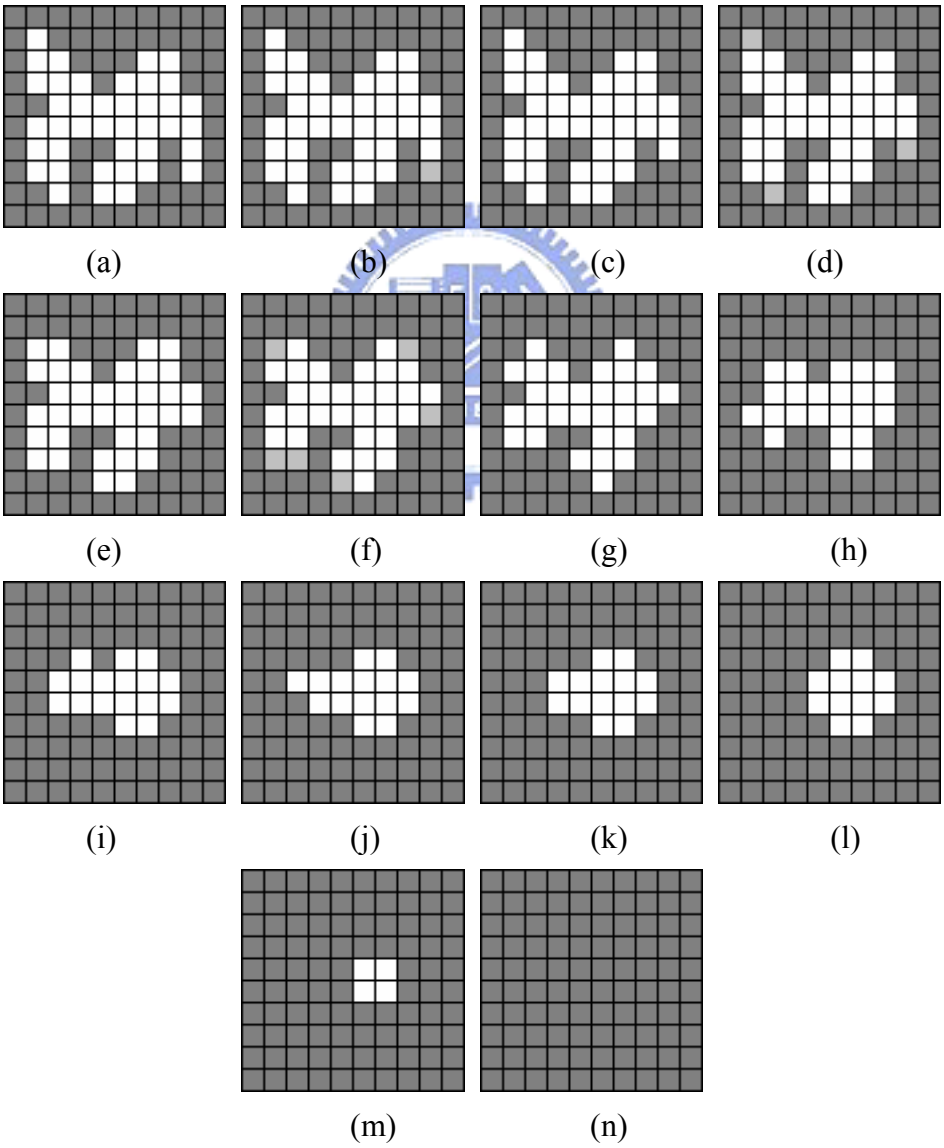


Figure 3.5 An example to demonstrate the elimination of a hue undefined region

3.3 Multilevel Thresholding by Hue Histogram

In this thesis, the angle of the hue is uniformly quantized into 256 parts. The histogram of hue is defined as $h(r_k) = n_k$, where r_k is the kth hue value and n_k is the number of pixels in the image with this kth hue value. After the hue histogram is computed, it can be normalized by dividing each value by the maxima.

Modes in the hue histogram represent the dominant colors for the image. Thus, to segment an image into regions corresponding to these dominant colors is equivalent to assign each pixel in the image to one of these modes. This can be achieved by multilevel thresholding the hue histogram. First, to determine the boundaries of the modes in the hue histogram, a peak finding algorithm proposed in [12] is used:

(1) Search local maxima in the histogram for the mode candidates.

(2) Find the local maxima again among those mode candidates to eliminate spurious candidates caused by noise.

(3) Eliminate candidates that are too small, too close, and not obvious. Experiments show that if we drop the candidates with value less than a threshold, say 0.05, then better results of color segmentation would be obtained. On the other hand, two modes are not expected to be close because of the similarity of the dominant hue value. Finally, the candidate is also not expected as mode if it is not very obvious. That is, if a candidate doesn't look

like a peak, it is not considered as a mode. Mathematically, whether the peak is obvious is determined by comparing the average value of the two candidate, $(h(p_1)+h(p_2))/2$, and that of the points between the two candidate, $h_{avg}=\frac{\sum_{p_i=p_1}^{p_i=p_2}h(p_i)}{p_2-p_1+1}$. Thus, the smaller candidate is dropped if $h_{avg}/((h(p_1)+h(p_2))/2)>0.75$. This is illustrated in fig. 3.6. The candidate of the two (red dotted) with smaller value will not be identified as a mode and only one mode is in this histogram.

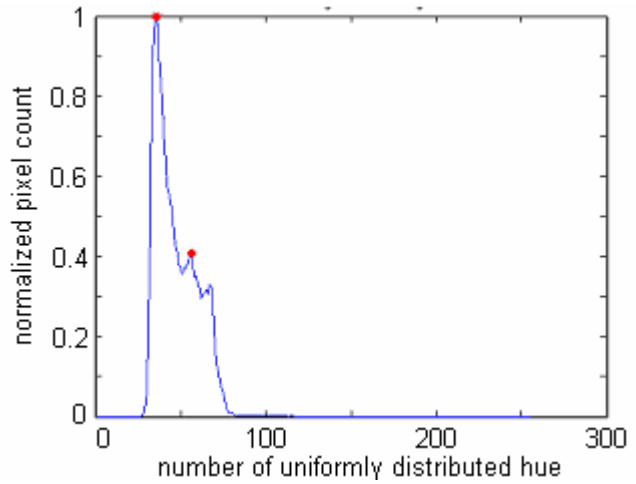
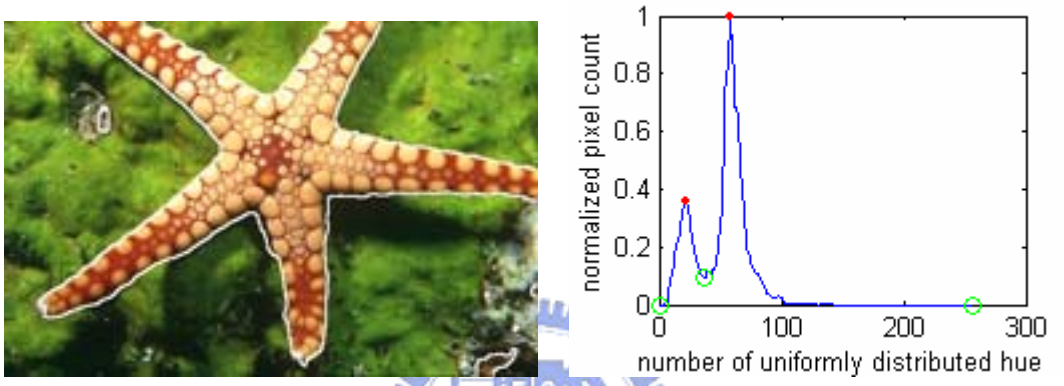


Figure 3.6 An example of the peak finding algorithm

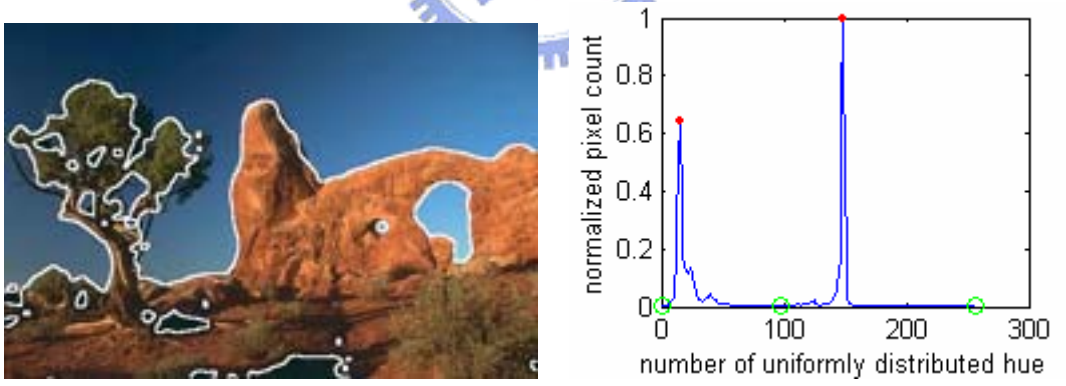
After the determination of peaks of the modes, the boundaries of the modes can be respectively chosen as the positions with minima values between the peaks. Now multilevel thresholding can be performed by the determined boundaries of the modes in the hue histogram. Each pixel with hue value between two adjacent boundaries is assigned to the corresponding dominant color. Fig 3.7(a) and 3.8(a) show the result

after hue undefined region elimination and multilevel thresholding the hue histogram, shown in fig. 3.7(b) and 3.8(b). In each hue histogram, the red dots mark the peak of each mode after the peak finding algorithm and the green circles mark the boundaries of the modes or the thresholds for multilevel thresholding.



(a) (b)

Figure 3.7 (a) Result after multilevel thresholding and (b) the hue histogram of (a)



(a) (b)

Figure 3.8 (a) Result after multilevel thresholding and (b) the hue histogram of (a)

Multilevel thresholding treats pixels with hue values within a certain range as a class. In the best case, the pixels of one class might tend to form one region corresponding to one object. However, these pixels might often be distributed into

two or more regions respectively corresponding to objects in the image. Moreover, in the worst case, these pixels might be grouped into several small regions which respectively correspond to some part of the interior of some objects as shown in fig. 3.8(a). This is because that histogram is a statistical operation that accounts for the whole image, but it lacks for consideration to the spatial information. As a result, it is often the case that, after multilevel thresholding the hue histogram, many small fractional regions might appear as in fig. 3.7(a) and 3.8(a). A complement to the pure global characteristic of histogram is to take spatial information into account [14]. The small fractional parts obtained by multilevel thresholding will be treated as fault segmentation and be merged to the adjacent region with the closest average of hue. The results of adding this spatial information to the segmentation result in fig. 3.7(a) and 3.8(a) are shown in fig. 3.9(a) and (b), respectively.

(a) (b)

Figure 3.9 (a) and (b) are results after merging small regions to their most similar adjacent regions

Chapter 4

Segmentation by Texture Cue

After the segmentation by color cue, several coarse regions had been generated. Ideally, each coarse region would contain only one non-textured or textured object with similar color, and no more segmentation for that region is necessary. But in the case that two or more objects with similar color yet distinguishable textures are segmented to one region after the first stage segmentation, they are expected to be further segmented. This leads to the need of the second stage to segment one region into two or more with different textures.

It is hard to define exactly what texture is. Usually, texture can be thought as the orderly pattern that looks like large numbers of small objects [1]. In this thesis, texture is treated as the pattern that comprises high density of orderly arranged edges.

4.1 The Filter Bank for Human Receptive Field

It has been proposed that there are cells in primate visual system which has similar operations as those commonly used in image processing [15]. Dots and lines of different orientations, lengths, and moving directions could respectively cause different responses to these cells; that is, for instance, a line at a certain orientation

and scale may result in a strong response to one cell but weak to another. Moreover, the operation corresponding to such cells is analogous to convolving the image with a certain linear filter. The models of these filters can be classified into three categories [7], [15]: filters that are radially symmetric, filters that are oriented even-symmetric, and filters that are oriented odd-symmetric. The radially symmetric filter is usually a Laplacian of Gaussian (LOG) or a Difference of Gaussian (DOG), which is an effective approximation to LOG. [16], [17]. The even-symmetric filter is usually an oriented Gaussian derivative. For example, the horizontal version of the filter is in the form:

$$f_{0^o}(x,y) = \frac{d^2}{dy^2}\left(\frac{1}{C}\exp\left(\frac{y^2}{\sigma_y^2}\right)\exp\left(\frac{x^2}{\sigma_x^2}\right)\right) \qquad (4\text{-}1)$$

where C is a normalization constant and σ_x and σ_y determine the scale of the filter in the x and y direction, respectively. Other even-symmetric filters are rotated copies of the horizontal even-symmetric filter. Finally, the oriented odd-symmetric filters can be modeled as the Hilbert transform of each line parallel to the direction of the even-symmetric filters.

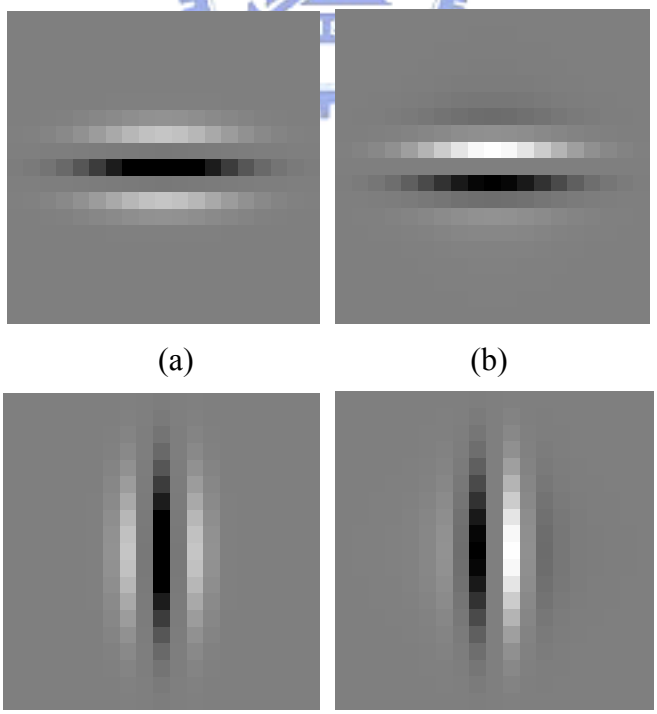
Equation (4-1) can be viewed as a Gaussian low pass function differentiated two times in one direction. The Fourier transform of a Gaussian function with standard deviation σ is still a Gaussian function with standard deviation $1/\sigma$. Therefore, Gaussian kernel can be treated as a low pass filter. Because both derivative and

convolution are linear operation, they satisfy the communitative property. As a result, the convolution of the image with the filter described in equation (4-1) is the same as convolving the image first with the Gaussian low pass filter and then differentiating the result. This can be qualitatively viewed as first denoising the image with a Gaussian low pass filter and this blurred image is then applied by a second derivative to find the gradient change in the image. Therefore, to denoise the image and find the change of gradient can be done by just one convolution of the image with the filter described by equation (4-1). From the viewpoint of frequency characteristics, due to the fact that differentiation enhances the high frequency signal, the combination of the Gaussian low pass filter and differentiation will form an effect of band pass filtering. Therefore, the oriented filter can be treated as a band pass filters for one direction and a low pass function for the other perpendicular direction. As a summary, the radially, even, and odd symmetric filters can be viewed as isotropic and oriented band pass filters.

4.2 Texture Analysis

Texture is a pattern with periodic characteristics. In other words, a texture region tends to consist of signals within a certain frequency band. In addition, slow change signals are usually caused by non-uniform illumination. Therefore, based on these two

basic observations, band pass filters can be reasonably used for texture analysis. However, although a large set of filters had been used in texture analysis [7] [11], to handle vectors which are constituted by the response of these filters in such a large dimension vector space is not very easy. As a result, only filters with orientations of 0 and 90 degree are used in this thesis to distinguish the textured and untextured regions. Fig. 4.1 shows the filters used in this thesis where the mid-gray level represents zero and darker gray level represents negative value while lighter gray level represents positive value. Fig. 4.1(a) and (b) are respectively the even- and odd-symmetric filters in horizontal direction, and (c) and (d) in vertical direction.



(a) (b)

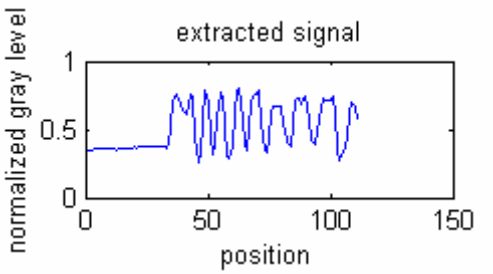
(c) (d)

Figure 4.1 (a) and (b) are the even and odd symmetric filters in the horizontal direction, and (c) and (d) in the vertical direction

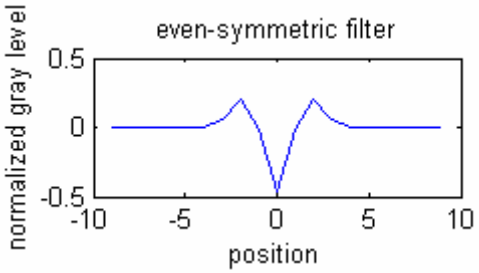
Assume that a region segmented in the first stage consists of textured and untextured patterns. The periodic textured pattern in the region is mainly composed of band pass signals, and band pass filtering will retain the characteristics of the texture. This remaining band pass signal can be treated as a clue to distinguish the textured pattern from the untextured one. To make the clue clearer and easier to use, the responses obtained by the band pass filters are absolutely added to form hills respectively corresponding to texture in the image. And this result is further low pass filtered to get the texture feature. To be a summary, the region is first filtered with the texture analysis filter bank, and the responses are absolutely added up and then low pass filtered to form the texture feature.

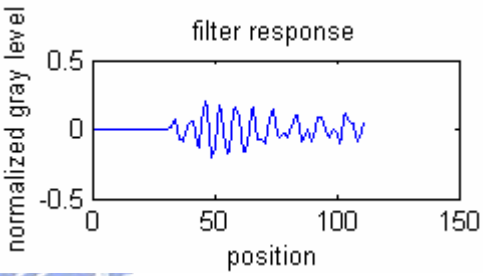
To illustrate this, an easy but comprehensive example is given first. Consider the one dimensional signal extracted from fig. 4.2(a), which is white labeled in (b) and shown in (c). Note that the signal is normalized to be in the range from 0 to 1. Then, after band pass filtering of fig. 4.2(c) with (d) and (f), only band pass signal will be retained as shown in (e) and (g), respectively. The absolute sum of the band pass signals (e) and (g) is shown in (h). Result of low pass filtering (h) is shown in (i). It is obvious that to classify the textured and untextured region is equivalent to distinguish the two plains with different height in fig. 4.2(i).

(a) (b)

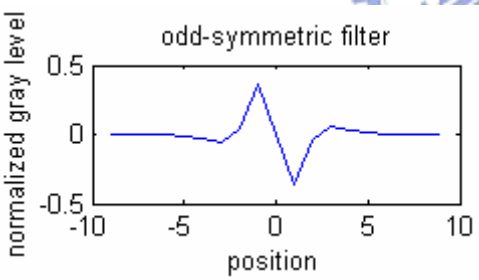


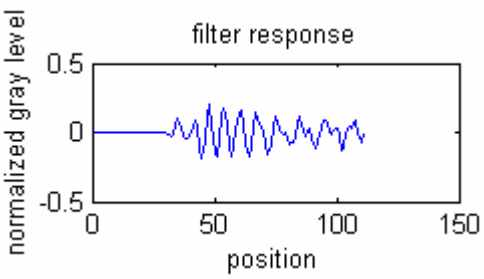
(c)



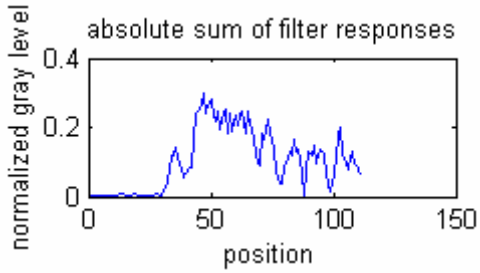


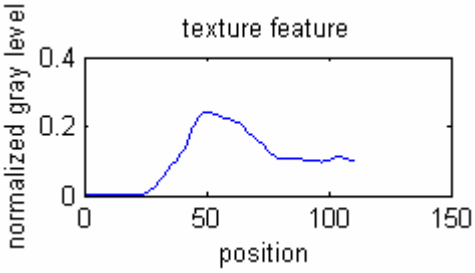
(d) (e)





(f) (g)

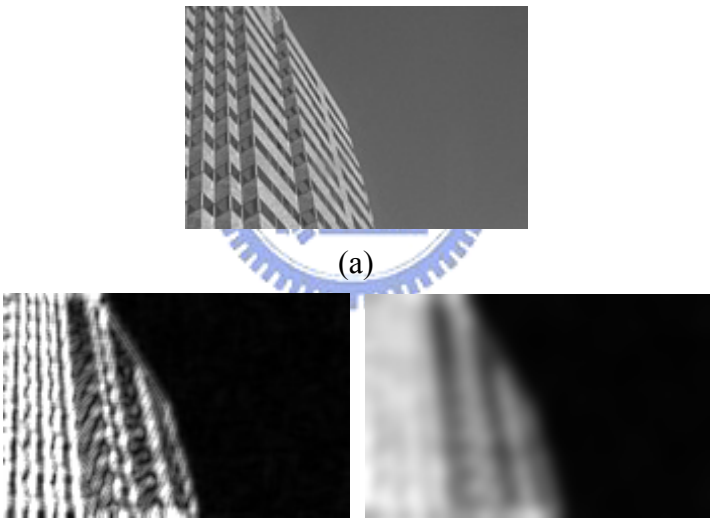




(h) (i)

Figure 4.2 An example of texture analysis

Now consider the whole image clip shown in fig. 4.2(a), which is reprinted in fig. 4.3(a). This image clip consists of a textured pattern composed of windows of a building on the left hand side and an untextured region caused by sky on the right hand side. Fig. 4.3(b) is the absolute sum of the responses of band pass filtering of (a) with the filters used in this thesis. The texture feature is the result of low pass filtering (b) with a Gaussian low pass filter and is shown in fig. 4.3(c). In fig. 4.3(c), regions with higher values correspond to the texture in the original image clip.



(a)

(b) (c)

Figure 4.3 The result of texture analysis shown as images

4.3 Region Merging

To determine the textured and untextured regions, the remaining task is to find regions with different average feature values in the result of texture analysis. The

pixels within the regions of higher average values correspond possibly to textured regions in the image. However, texture is usually not composed of a signal of single frequency for natural objects. Instead, in addition to the main frequency of the texture, signals of other frequencies comprise the texture and, as a result, some pixels within the textured region would possess values quit different to the average feature values of this region. Therefore, to find out such regions with high average feature values, an adaptive region merging method is adopted here [6]. This method is constituted by two parts: First, in place of seeds determination, the order of merging is determined, i.e. the order to check if two regions are to be merged is first obtained. Second, a merging predicate to check the merging of two regions is developed.

4.3.1 Order of Merging

The order of merging can play a significant role in region merging. Traditionally, the order of merging is just to scan an image, say, from top to bottom, left to right. As demonstrated in [6], the order of merging can be deliberately arranged such that better results can be obtained than the conventional order of merging. It is shown that bad ordering would result in wrong region merging even for images with little noise; yet good ordering is capable of tackling the noise in the image and helps to produce a good result.

Assume true regions are regions in the final segmentation result. A rule for the order of merging is: when any merging test between two (parts of) true regions occurs, all tests inside each of the two true regions have previously occurred. To simulate this order of merging, observe that, especially for non-textured regions, pixels in a true region tend to have low gradient and pixels near the boundary between true regions tend to have high gradient with respect to their adjacent pixels. Thus, the rule for the order of merging can be approximated by the increasing order of the gradient of adjacent pixels, i.e., check the merging predicate for pixels with low gradient first and then those with high gradient.

As an example for the order of merging, consider two adjacent pixels p and q with values p_v and q_v, respectively, the gradient of these two pixels is computed as:

$$f(p,q) = |p_v - q_v| \qquad (4\text{-}2)$$

For 4-adjacency, gradient of pixels are computed either in the horizontal or vertical directions using (4-2). For an image with height h and width w, h*(w-1) horizontal gradients and w*(h-1) vertical gradients can be computed. Then these gradients are sorted in the ascending order and this order is used as an approximation for the merging order.

4.3.2 Merging Predicate

To derive the merging predicate, first consider the novel model of image generation proposed in [6]. This model treats the image we have as an observed image I, which is sampled from a true image I*. In a true image, each pixel is represented by Q independent random variables. The sampling process of a pixel in the observed image from the true image is shown in fig. 4.4. Each pixel in the observed image is the sum of the outcomes of the Q independent random variables corresponding to that pixel in the true image. Assume each independent random variable takes positive values bounded by *g*/*Q*, then any possible sum of outcomes of the Q random variables is bounded by g and each pixel in the observed image has value no more than g.

To continue the derivation of the merging predicate, consider the following theorem:

Theorem 4.1 [6]:

Let $X = (X_1, X_2, ..., X_n)$ be a family of n independent random variables with X_k taking values in a set A_k for each *k*. Suppose that the real-valued function *f* defined on $\Pi_k A_k$ satisfies $|f(x) - f(x')| \le c_k$ whenever vectors x and x' differ only in the *k*th coordinate. Let μ be the expected value of the random variable $f(X)$. Then, for any $\tau \ge 0$,

$$\Pr(f(X) - \mu \ge \tau) \le \exp\left(-2\tau^2 / \sum_k (c_k)^2\right) \qquad (4\text{-}3)$$

Let $|R|$ be the number of pixel in region R. Then, for two regions R and R', there are $Q(|R|+|R'|)$ independent random variables. If a change of value of only one random variable that belongs to R occurs, the change of value of $|R-R'|$ will be bounded to $c_R = g/(Q|R|)$; if a change of value of only one random variable that belongs to R', the change of value of $|R-R'|$ will be bounded to $c_{R'} = g/(Q|R'|)$. Therefore, $\sum_k (c_k)^2 = Q(|R|(c_R)^2 + |R'|(c_{R'})^2) = (g^2/Q)(1/|R| + 1/|R'|)$. Let X be the vector constituted by the $Q(|R|+|R'|)$ random variables and f be defined as $f(X) = \overline{R} - \overline{R}'$, where $\overline{R}$ is the average values of observed pixels with in R. Substitute these into (4-3) and using the fact that the deviation with the absolute value is at most twice that without, then

$$\Pr\left(\left|(\overline{R}-\overline{R}') - E(\overline{R}-\overline{R}')\right| \geq \tau\right)$$

$$\leq 2\exp\left(-2\tau^2/\left(g^2/Q\right)\left(1/|R|+1/|R'|\right)\right) = \delta \qquad (4\text{-}4)$$

If δ is chosen, then τ can be determined by

$$2\exp\left(-2\tau^2/\left(g^2/Q\right)\left(1/|R|+1/|R'|\right)\right) = \delta$$

$$\Rightarrow \quad \tau = g\sqrt{\frac{1}{2Q}\left(\frac{1}{|R|}+\frac{1}{|R'|}\right)\ln\frac{2}{\delta}} \qquad (4\text{-}5)$$

In other words, we have, with probability no more than δ,

$$\left|(\overline{R}-\overline{R}') - E(\overline{R}-\overline{R}')\right| \geq g\sqrt{\frac{1}{2Q}\left(\frac{1}{|R|}+\frac{1}{|R'|}\right)\ln\frac{2}{\delta}} \qquad (4\text{-}6)$$

This leads to the following corollary.

Corollary 4.1:

Consider a fixed couple (R, R') of regions of I., $\forall 0 < \delta \le 1$, the probability is no more than δ that

$$\left|(\bar{R} - \bar{R}') - E(\bar{R} - \bar{R}')\right| \ge g\sqrt{\frac{1}{2Q}\left(\frac{1}{|R|} + \frac{1}{|R'|}\right)\ln\frac{2}{\delta}}$$

Because that each random variable is independent, all couples of regions (R, R') whose merging are tested will satisfy $\left|(\bar{R} - \bar{R}') - E(\bar{R} - \bar{R}')\right| \le g\sqrt{\frac{1}{2Q}\left(\frac{1}{|R|} + \frac{1}{|R'|}\right)\ln\frac{2}{\delta}}$ with a probability $\ge (1-\delta)^N$. If δ is small, then $(1-\delta)^N \approx 1 - N\delta$. When pixels of $R \cup R'$ actually come from the same statistical region in the true image I*, then $E(\bar{R} - \bar{R}') = 0$. Thus, it can be summarized as that $\bar{R} - \bar{R}'$ does not exceed $g\sqrt{\frac{1}{2Q}\left(\frac{1}{|R|} + \frac{1}{|R'|}\right)\ln\frac{2}{\delta}}$ with a high probability. This is the merging predicate used in for region merging. It is summarized as following:

$$P(R, R') = \begin{cases} true & if \quad \left|(\bar{R} - \bar{R}')\right| \le g\sqrt{\frac{1}{2Q}\left(\frac{1}{|R|} + \frac{1}{|R'|}\right)\ln\frac{2}{\delta}} \\ false & otherwise \end{cases} \qquad (4\text{-}7)$$

Qualitatively, the merging predicate can be treated as an adaptive merging rule. If two small regions are tested by the merging predicate, the threshold in the right hand side of (4-7) will be large and these two regions will be merged to one as long as the difference between the average values of these two regions is not too much. On the other hand, two large regions only merged when their average values are very similar.

For example, by the above order of merging, most pixels within the left and right hand side regions in fig. 4.3(c) will be first checked and merged because their values are similar to their adjacent pixels. Although some pixels with different feature values will appear within the two regions, they will be merged due to high threshold because these pixels are scattered separately within the regions. This can be viewed as a denoising property of this region merging method. Fig. 4.5 shows the result of applying this region merging method to fig. 4.3(c).

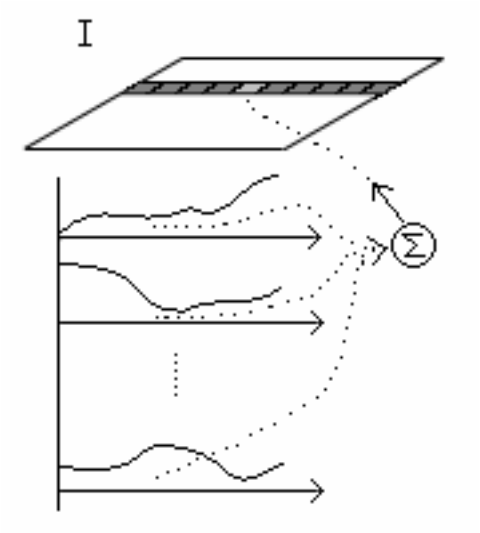
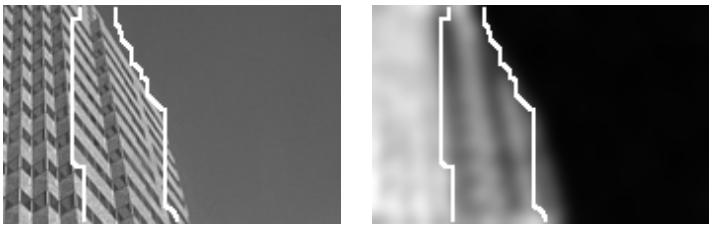


Figure 4.4 The model of image generation



(a) (b)

Figure 4.5 Results of region merging after texture analysis

Chapter 5

Experimental Results

The test images used in the experiments are taken form the Berkeley segmentation dataset [18] with sizes of either 481x321 or 321x481. Experiments are done for these images to determine the parameters of the method in this thesis. The standard deviation of the low pass Gaussian denoising filter is set to $\sigma = 2$. Regions that are less than 2.5 percent of the total number of the pixels in the image are merged to their most similar adjacent region after multilevel thresholding of the hue histogram. The scales in each of the band pass filters are 1 and 3 corresponding to the direction of the filters. The standard deviation of the Gaussian low pass filters for texture analysis is set to 4. And, finally, the parameters for the region merging are set to g=255, Q=8, and $\delta = \frac{1}{(h * w)^2}$, where *h* and *w* are respectively the height and width of the input image. The following experiments are done based on these parameter settings if not mentioned.

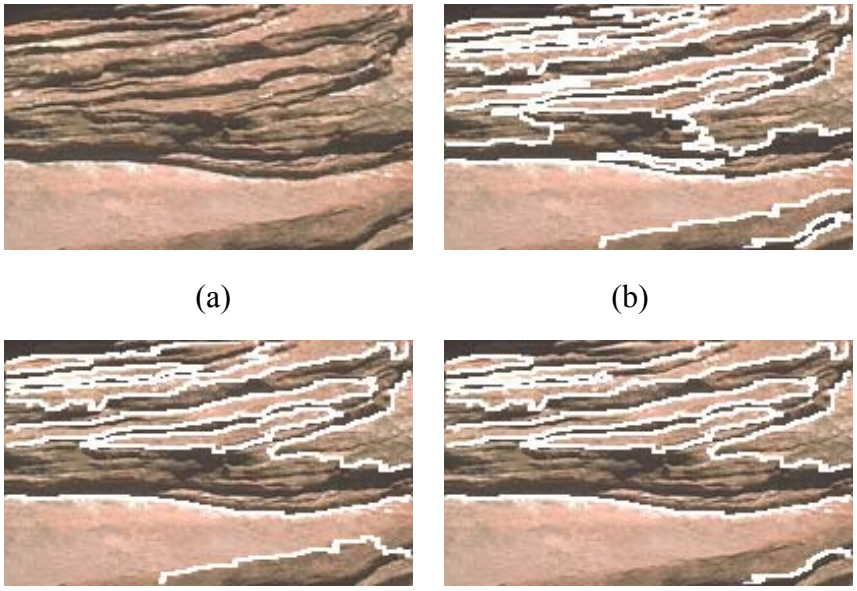
5.1 Experiments for Texture Segmentation

Some experiments for the texture segmentation are shown in this section. It has been discussed that conventional region merging methods or even the region merging

method used in this thesis are more suitable for untextured image segmentation. Due to the lack of the information of texture, a texture region is often improperly segmented to many regions using region merging. Fig. 5.1(a) shows an image clip composed of two regions with similar color but different texture at the upper and bottom side, respectively. Fig. 5.1(b), (c), and (d) are the results of applying directly the region merging method introduced in section 4.3 to fig. 5.1(a), respectively with Q=4, Q=2, and Q=1. It is obvious that the upper texture region tends to be segmented to several regions each of which is of an average brightness value different to those of its adjacent regions. Obviously, direct utilization of region merging to separate two regions with different texture is difficult.

On the other hand, the pixels within the textured region reveal their common feature after texture analysis. And region merging based on this texture feature would produce a better segmentation of textured and untextured regions than that obtained by direct use of the region merging. Fig. 5.2 shows some results of region merging for the texture feature. Clearly, region merging for the texture feature improves the performance of the segmentation for textured and untextured regions. Other examples are shown in fig. 5.3 and 5.4 to demonstrate the resistance of the texture segmentation in this thesis to the influence of shadow. Note that in fig. 5.3, the textured and untextured regions are segmented mainly due to the darker region between them

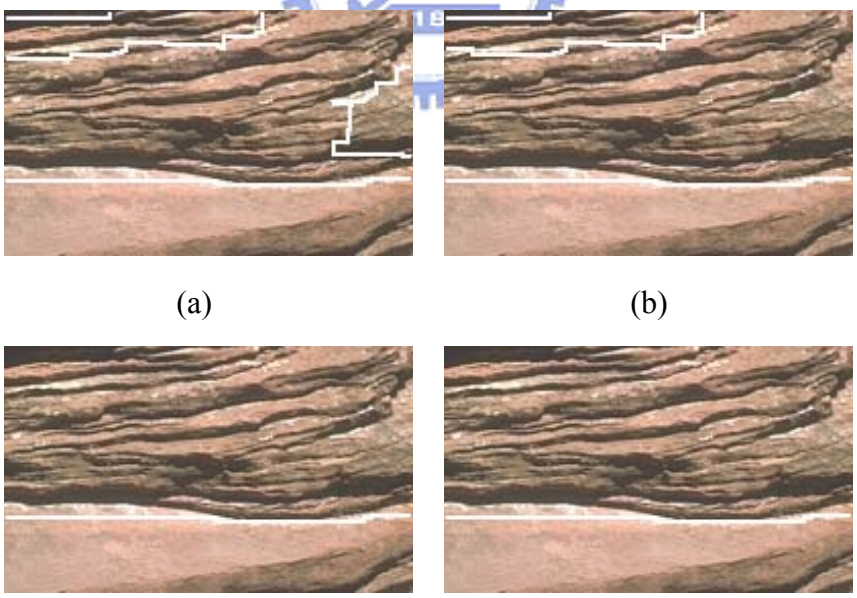
caused by shadow.



(a) (b)

(c) (d)

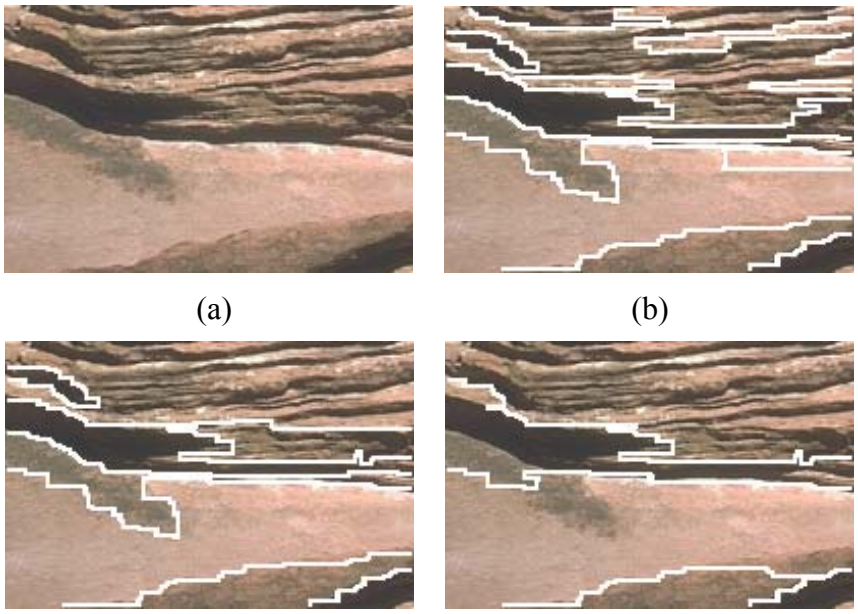
Figure 5.1 (a) An image clip and the results of region merging with (b) Q=4 and (c) Q=2 and (d) Q=1



(a) (b)

(c) (d)

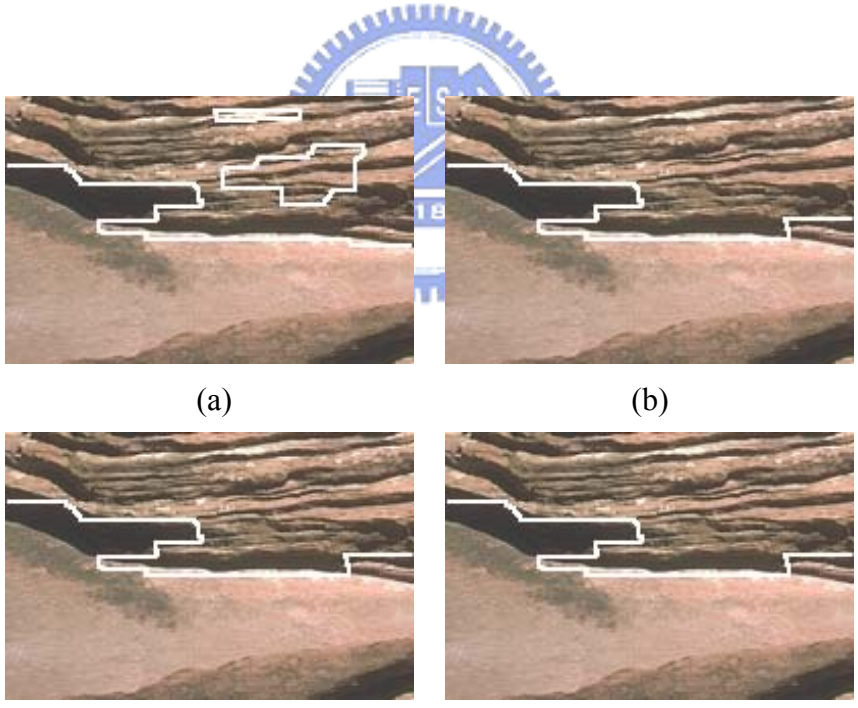
Figure 5.2 Results of the region merging after texture analysis with (a) Q=8, (b) Q=4, (c) Q=2, (d) Q=1



(a) (b)

(c) (d)

Figure 5.3 (a) An image clip and the results of region merging with (b) Q=4 and (c) Q=2 and (d) Q=1



(a) (b)

(c) (d)

Figure 5.4 Results of the region merging after texture analysis with (a) Q=8, (b) Q=4, (c) Q=2, (d) Q=1

Generally speaking, the segmentation in one of the stages in the hierarchical segmentation method can take advantage of the segmentation result of its preceding stage. Because only a two-stage-segmentation is used in this thesis, experiments will be done for the textured segmentation with and without the information given by the color segmentation. If the texture segmentation is applied to the texture feature of the whole image in fig. 5.5(a), then the segmentation results are obtained in fig. 5.5(b), and (c). These results can be compared to that shown in fig. 5.5 (d), which is the result of texture segmentation based on the result of color segmentation. The improvement of the performance by using the hierarchical segmentation is evident.

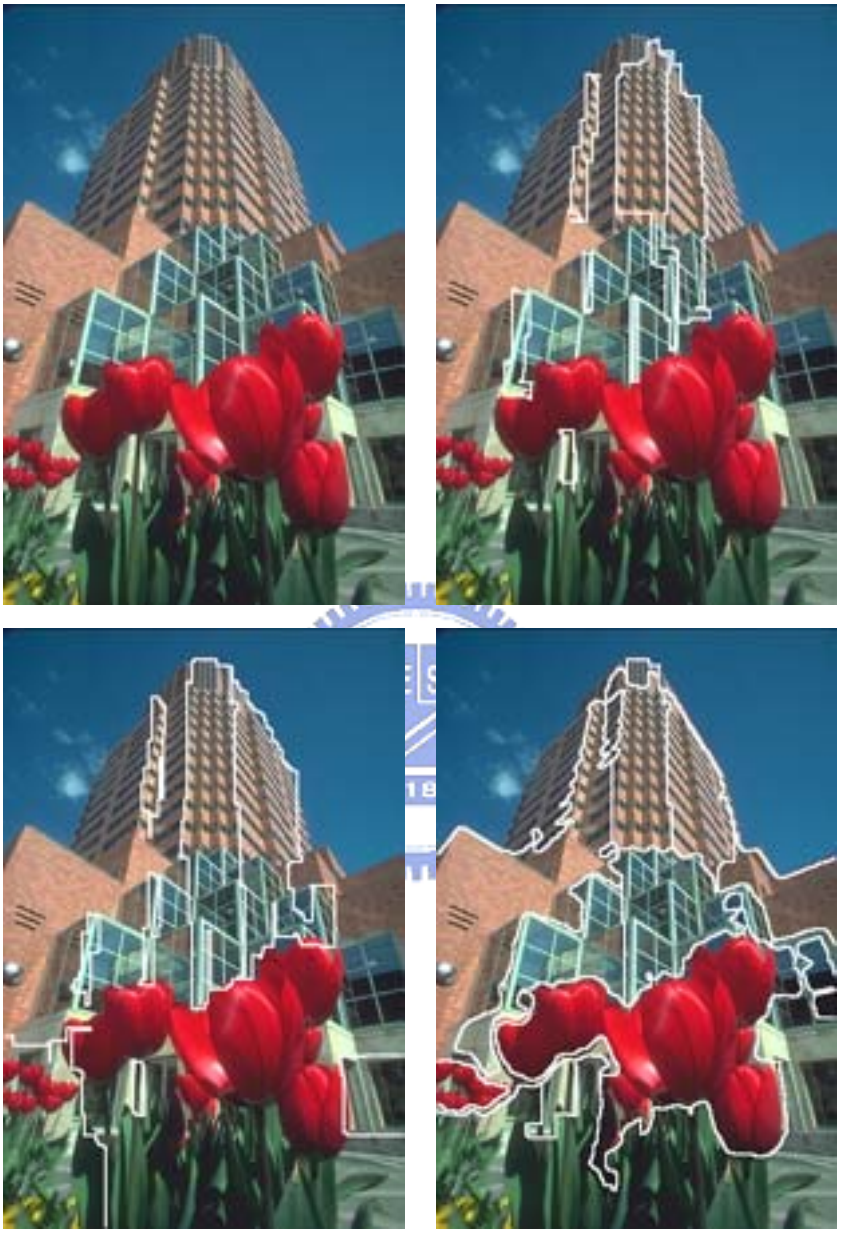
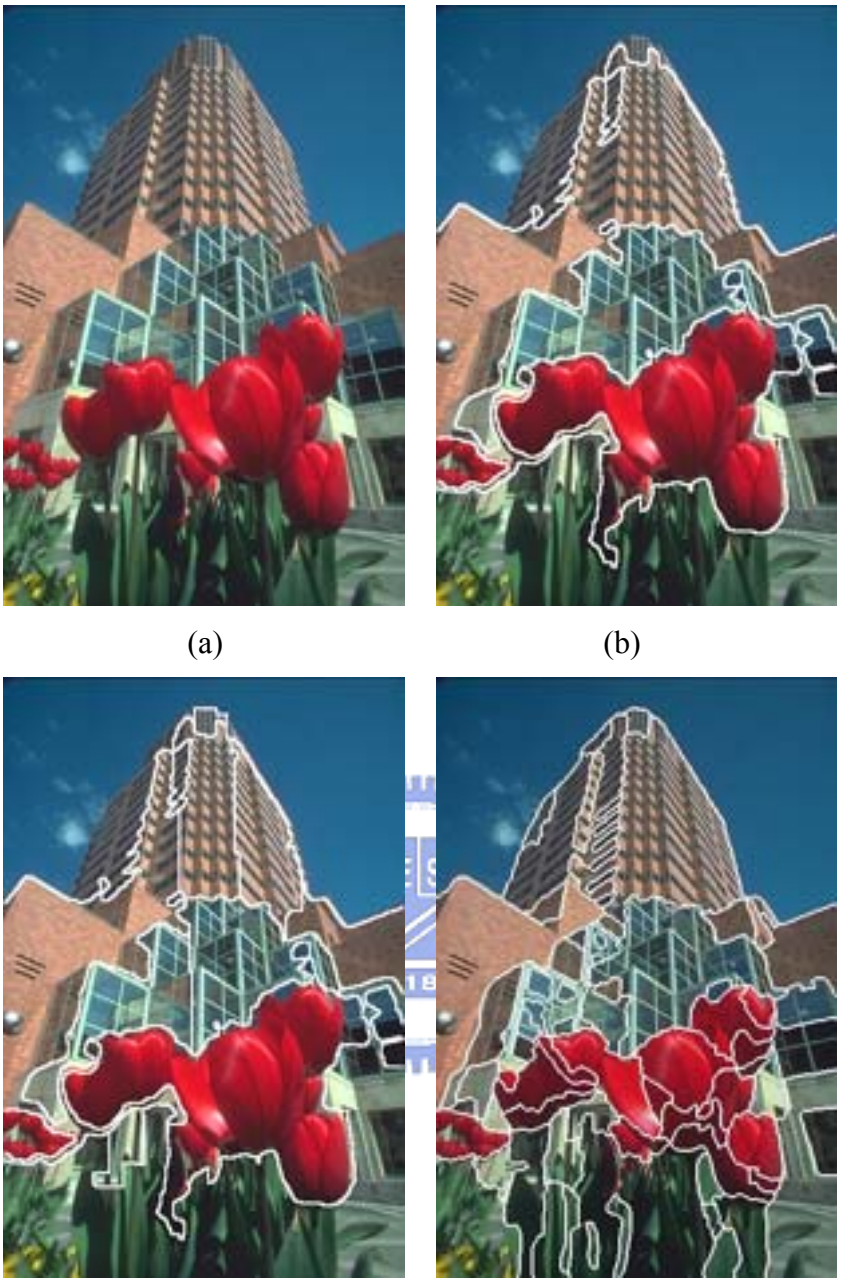


Figure 5.5 (a) Original image and results of texture segmentation with (b) Q=8 and (c) Q=4 to the whole image, and (d) a result of our segmentation method with Q=8

5.2 Comparison of the Segmentation Results

The image segmentation method in this thesis is compared with a segmentation method (JSEG) which segments color-texture regions in an image [19]. This method uses color quantization and spatial segmentation to achieve the segmentation task. By the inherence of JSEG, its performance is heavily influenced by the spatially varying illumination, and thus an object with same color would be segmented to several regions due to the influence of non-uniform illumination. Fig 5.6 and 5.7 show results of our one-stage and two-stage segmentation, and the results obtained by JSEG. It can be seen that the JSEG segments the flower in fig 5.6(a) and the sky in fig 5.7(a) into several regions due to non-uniform illumination while the results obtained by the segmentation method in this thesis retain the completeness of color objects. Note that the results of JSEG are obtained with its default parameter setting.

Experiments are also done for some car images as shown in fig. 5.8. The results of one-stage and two-stage segmentation are shown respectively in the second and third row in fig. 5.8. It is believed that even if only the results of one-stage segmentation are obtained, the information can still facilitate the searching of license plate. For the cases shown in fig. 5.8, license plates are successfully found in the results of the two-stage segmentation.



(a) (b)

(c) (d)

Figure 5.6 (a) The original image, (b) the result after 1st stage segmentation, (c) the result after 2nd stage segmentation, and (d) a result obtained by JSEG

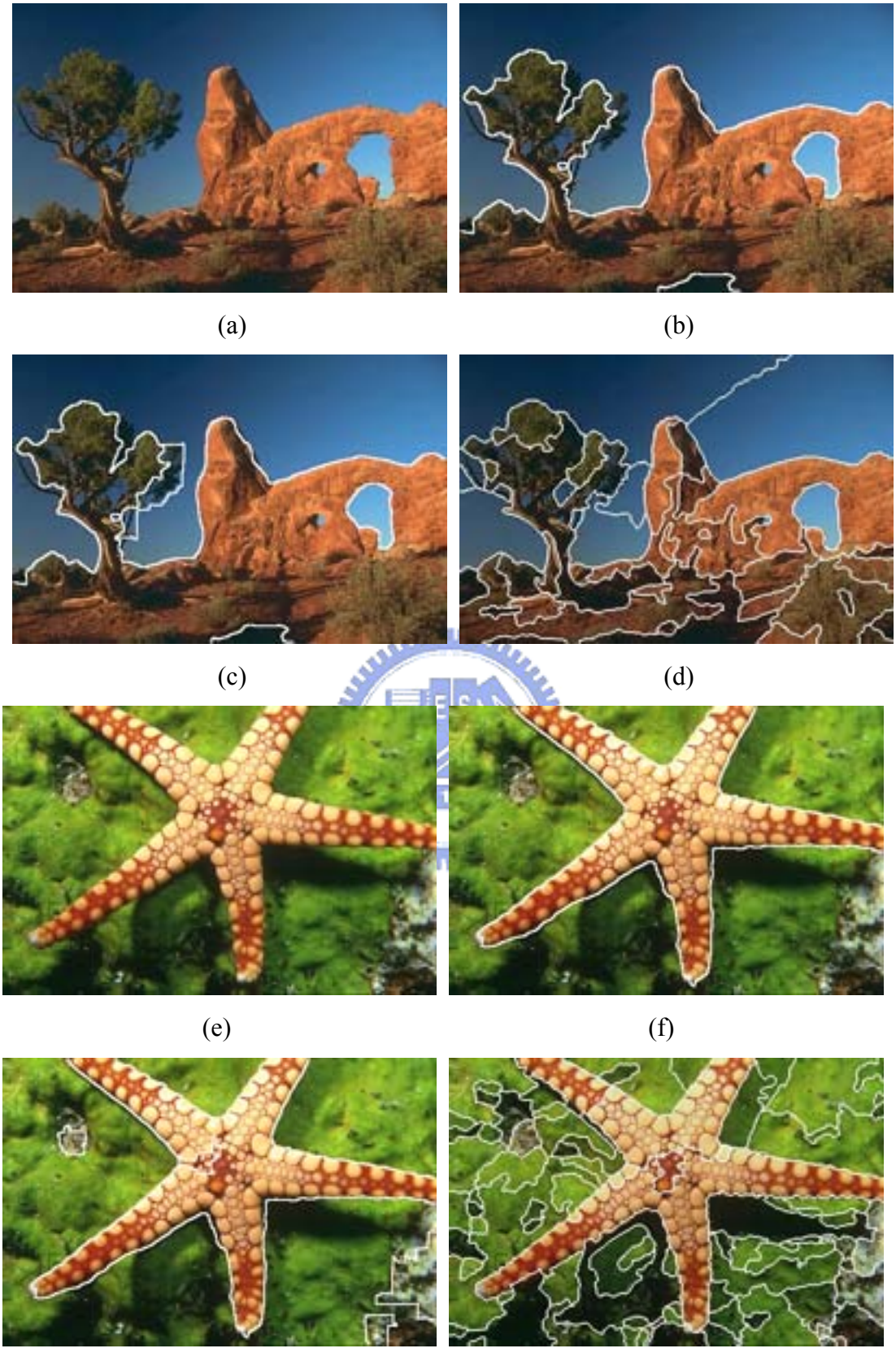


Figure 5.7 (a)(e) The original images, (b)(f) the results after 1st stage segmentation, (c)(g) the results after 2nd stage segmentation, and (d)(h) results obtained by JSEG

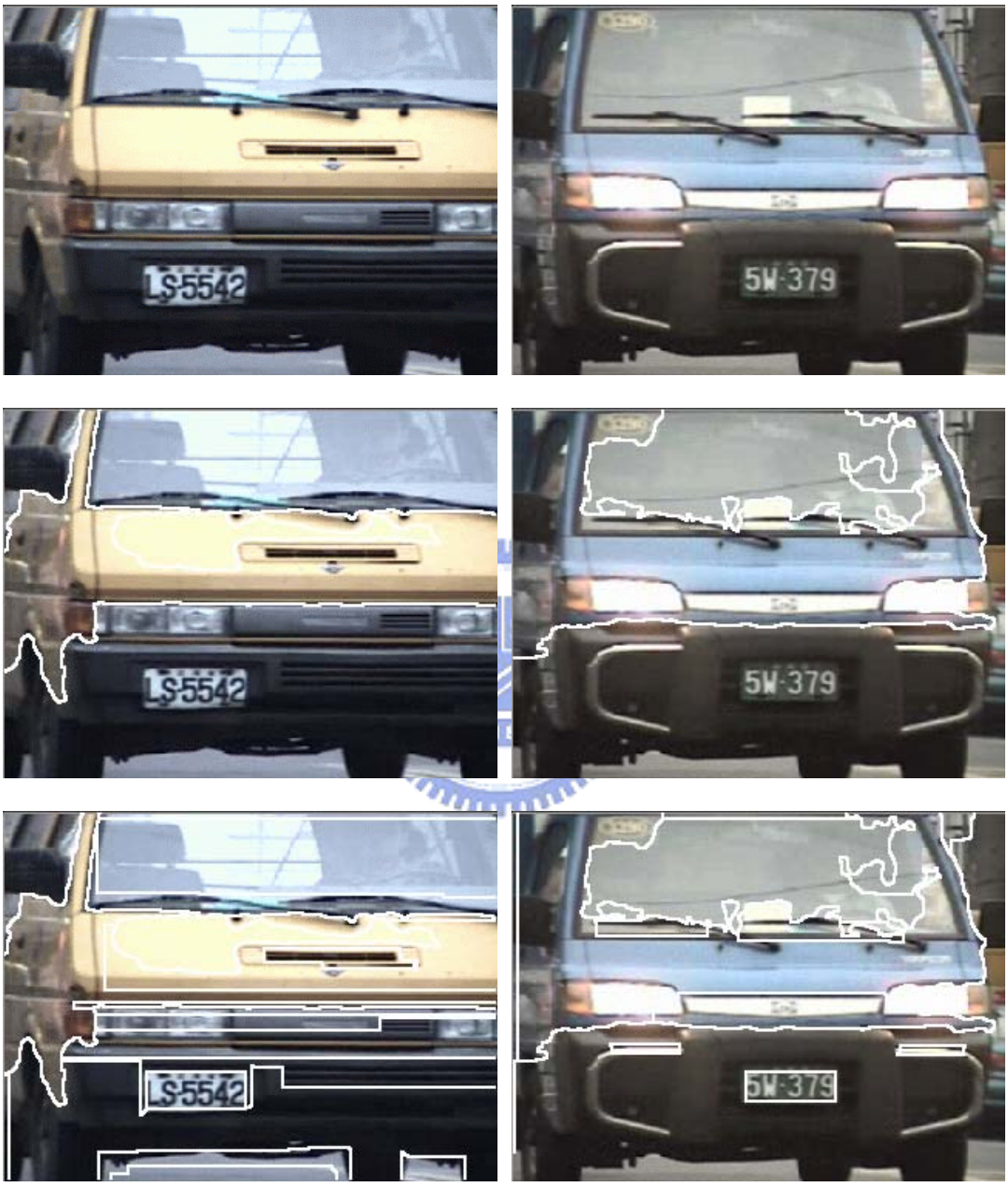


Figure 5.8 Results of image segmentation for car images

Chapter 6

Conclusions and Future Works

A hierarchical scheme for natural image segmentation is proposed in this thesis. This thesis demonstrates a segmentation scheme using color and texture at different stage to hierarchically segment an image into regions with similar color and texture. Experiments show that the segmentation in the first stage by color hue has resistance to the influence of non-uniform illumination and tends to produce coarse or under-segmentation results for natural images. The regions obtained in the result of first stage segmentation have the advantage of purifying the information in each of the regions and, as a result, the second stage of segmentation can deal with these meaningful regions rather than the whole confused image. In addition, the second stage of segmentation has chance to selectively process the regions in the result of the first stage of segmentation by some simple examination. This can facilitate the speed or performance of the segmentation. The segmentation by texture in the second stage provides an easy and intuitive way to segment the textured and non-textured regions.

The separation of each cue in different segmentation stages can simplify the use of each cue. Besides, it is easy for the extension to append more segmentation stages based on other information. For example, if the range data of an image is obtainable,

then it can be used for the first or second stage segmentation, resulting in a three-stage segmentation method. Moreover, segmentation method in each of the stage can be easily replaced by others according to the need of users. Of course, wrong segmentation in one stage will mislead all the following segmentation. Therefore, the selection of information used in each segmentation stage is important and a rule might be established to detect if the regions in the preceding segmentation result are usable.

However, there are some inherent difficulties in this method. First, hue is not a uniform color space, i.e. the degree of the perception of human for the difference of two colors is not the same to the difference of their hue. As a result, two regions with distinguishable hue may look like regions with similar color. In addition, hue is not capable of dealing with black and white color and is not stable for dark brightness. For example, the black and white regions in the image shown in fig. 6.1 can not be segment perfectly by the use of only hue information. Thus, it is necessary to do some modification or improvement for the segmentation by color in the first stage to tackle the black or white regions in the image. Second, the result for texture segmentation can be heavily infected by the sizes and ratios of the band pass filters and the Q of region merging. Although these parameters have been set to some value, in fact, they may be set automatically according to the information of the images. For the segmentation method in this thesis, the parameters for the second stage segmentation

have chance to be adaptively and respectively set corresponding to each of the regions in the first stage.

Figure 6.1 Bad segmentation results for black-white regions