University of Alberta

On the use of CBIR in Image Mosaic Generation

by

Yue Zhang ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2002

National Library of Canada

Acquisitions and Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada

Acquisitions et services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-81508-0

Canada

University of Alberta

Library Release Form

Name of Author: Yue Zhang

Title of Thesis: On the use of CBIR in Image Mosaic Generation

Degree: Master of Science

Year this Degree Granted: 2002

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Zhang Yue

Yue Zhang
505-8515, 112 Street
Edmonton, Alberta
Canada, T6G 1K7

Date: July 17, 2002

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **On the use of CBIR in Image Mosaic Generation** submitted by Yue Zhang in partial fulfillment of the requirements for the degree of **Master of Science**.

[illegible]
Mario A. Nascimento

[illegible]
Osmar R. Zaïane

[illegible]
Davood Rafiei

[illegible]
John Freeman

Date: July 16, 2002

Abstract

Image mosaic or image montage is an image made up of many other images. An image mosaic has its own visual content as a whole while each of its building images also has a meaningful content. This thesis address the design and implementation of an image mosaic generating system, which uses computers to build image mosaics automatically. There are many parameters used in our system to control an image mosaic generating process which are shown playing different roles on affecting the processing time and image mosaic's quality.

As another contribution, a new approach used to evaluate an image mosaic's quality is proposed in this thesis, which is based on the human perception of image mosaics. It uses the increasing speed of distance between local colour histograms of an image mosaic and its original image as the distance measure. And the experimental results show that this distance measure performs the most stable for evaluating image mosaics' quality in comparison with Global Colour Histogram and Average Pixel-to-Pixel distance measure.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisors, Dr. Mairo A. Nascimento and Dr. Osmar R. Zaïane. Their guidance and experience are priceless contributions to this research. They always encourage me and give me great suggestions during this research. They have spent much time and effort on this thesis research which deserves my special appreciation.

I am very grateful to my wife, who did what she can do to help me on this research. She spent much time discussing problems with me and gave me many suggestions. I would also like to thank my parents for their always encouragement and solicitude.

I am thankful to Anne Nield for proof-reading this thesis. I also would like to thank my colleagues in the database group, staff members and graduate students of the Department of Computing Science for providing such a friendly and enjoyable environment here. Finally, I gratefully acknowledge the use of the S-Tree source code provided by Eleni Tousidou.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

1.1 Introduction of Image Mosaics

Is there a relationship between computer science and the arts? Yes. Using computer technology in different kinds of art has a long history. Especially nowadays, the emergence of various advanced computer technologies gives computer art a whole new life. We have seen a broad range of work which is created using computers as mediums or tools. For instance, Echen uses different colours to denote different letters in forming his "Art from text" [6]. Marius Hartman explores facial multiplicity by combining different faces generated by a computer at different times in his "10000 zombies" [13]. Many other examples of computer art exist – ranging from digital music composed by computers to virtual cameras used in the 3D world.

Robert Silvers, a former MIT media Lab graduate student, used a computer as a tool to create images with an amazing appearance, called photomosaics [27]. A photomosaic, or an image mosaic, or an image montage – which is also the focus of our thesis – is an image made up of many other images. Each of the building images has its own content, and all these building images are put together to generate a single image which also has its own meaningful content. The visual effect of an image mosaic is that its content, instead of each building image, can only be seen clearly from many steps away.

Using many images to build a single image has a long history. Back in the

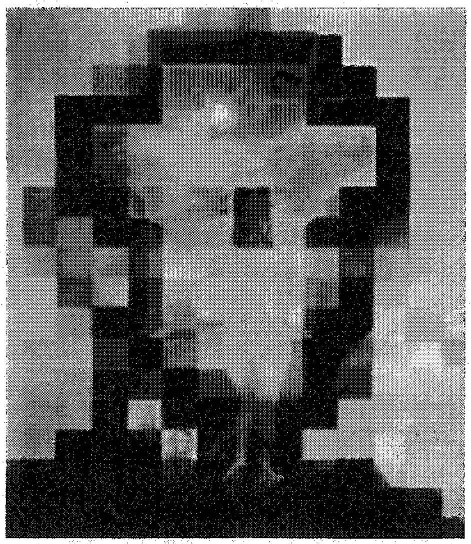


Figure 1.1: The image montage of Abraham Lincoln [Salvador Dali]

16th century, the painter Giuseppe Arcimboldo used vegetables, fruit, and flowers as building images to build an image mosaic [37]. Salvador Dali created a well known image montage of Abraham Lincoln by putting many other images together including his wife's picture, which is shown in Figure 1.1. Using computers to build image mosaics is a big step in their history. Robert Silvers, as we mentioned, is the inventor of using computer technology to generate image mosaics, and the patent holder for his image mosaic generating system "Photomosaic" [27].

1.2 Image Mosaics and Image Retrieval

In general, an image mosaic is generated based on an original image. The requirement of the image mosaic is that it retains similar visual content to the original image, while being made up of many images. Most modern image mosaic generating systems using computer technology contain the following steps:

- Divide the original image into many tiles.
- For each tile, find another image with similar content from an image

database.

- Build the image mosaic by replacing all tiles by their similar images.

The second step – selecting an image having similar content with a tile from a large image database – is called Content-based Image Retrieval (CBIR). CBIR plays an important role in an image mosaic generating system. The quality of the resulting image mosaic directly depends on the effectiveness of the CBIR technique used in the system.

CBIR is currently an active research area in computing science, as more and more visual information – especially digital images – has been made available in digital archives. The basic task of a CBIR system is to find similar images according to their visual features, within a large image database. Query By Example (QBE) is one of the most popular methodologies used in CBIR systems, in which images are selected from an image database similar to a given image presented by users. IBM's QBIC [21] is such a CBIR system.

Typically, a CBIR system pre-processes an image database in order to extract information from all images in the database. This information is referred to as visual features of images. The visual feature may be represented by different visual feature representations, such as the image's colour histogram. Different CBIR systems can be categorized based on different visual features extracted, as well as different abstract visual feature representations used. When an example image is given, the same visual feature is extracted from this image and used to match all the visual features of images in the image database. Based on some distance metric, the image with the smallest distance to the given image is retrieved as the result. In current CBIR systems, instead of only retrieving the best image, a number of similar images are retrieved and sorted according to their distance to the given image. Since CBIR is the central part of an image mosaic generating system, in Chapter 2, the main theory of CBIR systems and related work will be discussed in detail.

1.3 Our Work – Mosaicture

Image mosaics provide an interesting topic for both computer scientists and artists. Some work and research has been done on this topic, though there are few methodologies and implementations involving image mosaic generating systems. In fact, not much is known even about the most famous Silvers' patent implementation. In addition, since the quality of image mosaics is based largely on human perception, it is difficult to use any quantitative method to evaluate their quality. This problem of subjectivity may be one reason that not much has been studied about this topic.

In our work, we propose a methodology of generating image mosaics. Our image mosaic generating system divides an input image into many tiles; and then for each tile, it fetches the image with the most similar content from an image database and replace the tile with the image. CBIR techniques and a approach we proposed are used in combination to select the image with the most similar content to a tile. Due to the large size of our image collection, a compact visual feature representation, as well as its distance metric, has been used to reduce the storage and processing time. There is a set of parameters used in our system, such as the number of quantization colours and the size of the image database, which control the procedure of image mosaic generating. Figure 1.2 shows one image mosaic generated by our system, as well as magnified parts of some areas.

Since there are few image mosaic generating methodologies and most implementations are not thoroughly studied, we have chosen not to focus on the comparison of our image mosaic generating methodology with others. Instead, we delve into our methodology and analyze the different roles and effects of a variety of factors in the system. Our conclusion, after analyzing a large number of experimental results, is that the number of quantization colours is the most critical factor and plays the most important role in our methodology. The size of the image database is another factor that affects the final result of an image mosaic. Other factors, such as how to use the colour feature representation as well as the approach to select the image with the most similar content to a tile, are also studied.

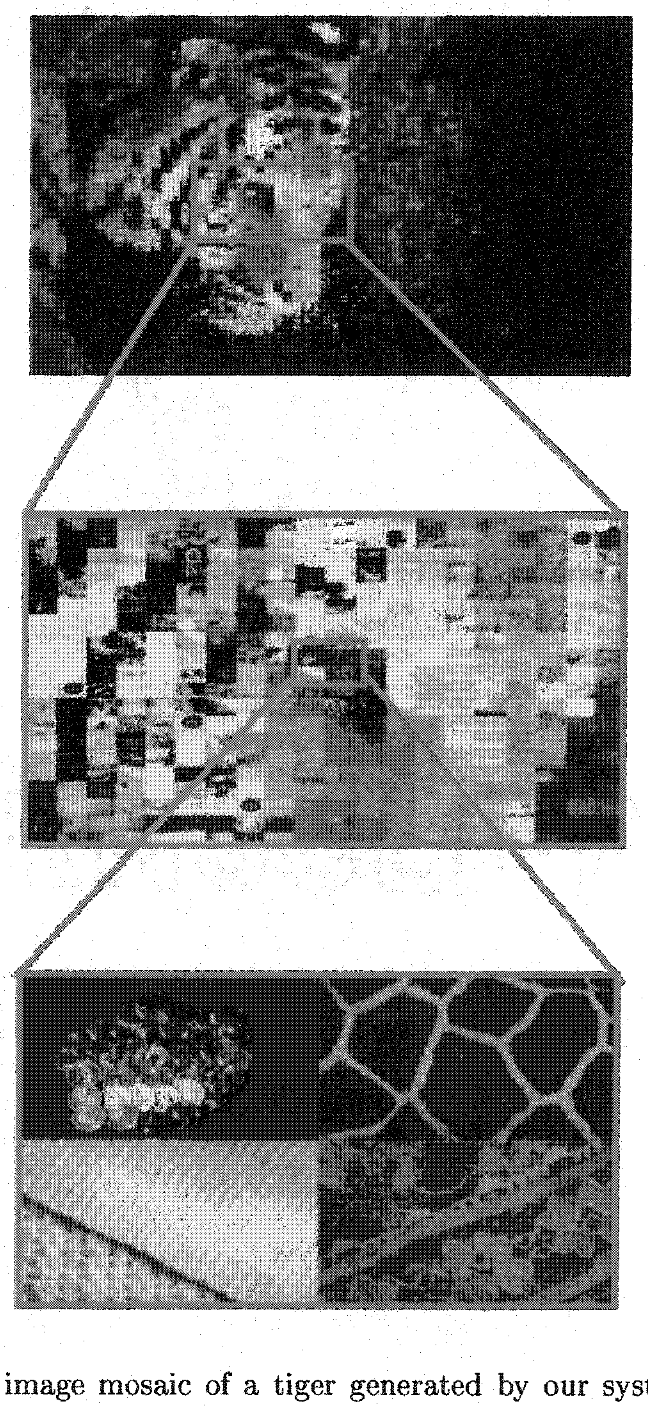


Figure 1.2: An image mosaic of a tiger generated by our system, which also shows detailed look at some areas

As mentioned earlier, evaluating the quality of an image mosaic is a subjective problem. There is currently no existing image mosaic generating system using any quantitative method to evaluate the quality of image mosaics. In our work, we attempt to use approaches such as pixel-by-pixel distance metric and histogram distance metric, as well as a new approach we propose to evaluate the quality of image mosaics in quantitative way.

1.4 Thesis organization

This thesis is organized as follows. Chapter 2 contains the basic information and main theory of CBIR systems, as well as its implementation and a small survey. Chapter 3 is the main contribution of our thesis. Introduction of image mosaic's history and related work is in the beginning part of that chapter. Our proposed methodology for generating image mosaics is described after that in detail as well as the discussion of the parameters used in our system. In addition, implementation issues are discussed in this chapter. Discussions about methods to evaluate the quality of an image mosaic are contained in Chapter 4. Also, all the experimental setups, results and analyses as well as the conclusions are included in that chapter. Chapter 5 concludes our work as well as presenting a discussion of future work.

Chapter 2

CBIR and Related work

2.1 Introduction of CBIR

The growth of multimedia information has been enormous recently, especially with the advent of the Internet. A huge digital image archive is made up of millions of images, photos created by hospitals, governments, companies and academic organizations. These images are useful in many fields. For instance, a large number of satellite photos can be used for weather forecasting purposes; X-ray photos of human bodies are useful in the medical field; images of human faces are critical for the police to identify criminals. However, we cannot access or make use of these images unless they are well organized and easily retrievable. Searching for the face of a specific person in millions of facial images is very tedious and frustrating. Originally, searching an image database was based on human annotation: each image in a database is given some keywords to denote the semantic content of the image; then, all the keywords are used to index images. Thus, searching and retrieving images is based on the keywords of images. This is called Text-based Image Retrieval [9]. This approach is easy to understand in comparison with other approaches we will be discussing later. However, as we can easily see, this approach has many limitations.

- As the size of image collections gets increasingly large, manually giving each image an annotation is impractical.

- Annotating an image based on human perception is subjective. Different people may give different annotations to images with similar visual contents.

In the early 1990s, Content-based Image Retrieval (CBIR) was proposed to overcome the limitations of Text-based Image Retrieval. In CBIR, images in a database are indexed using their own primitive visual features instead of human annotations such as shapes, colours, and textures. The use of different visual features is also a criterion to categorize a CBIR system. Since the visual features of an image are only based on the image itself, there is no problem of subjectivity. When an example image is given, its visual features are also extracted and used to match against those in the database. Some distance metrics are used to compute the similarity between the query image and images in the database. The result of the query is a set of images similar to the query image, rather than an exact match. These result images are also sorted according to their distance to the query image. Visual feature extraction, distance metric, and different CBIR techniques will be discussed in detail in the following sections. However, CBIR also has its own limitations. The main problem is that it cannot deal with semantic-level image queries effectively. An image always contains some semantic information (for example, an image containing a little boy). Such semantic features can only be represented by some primitive features in present CBIR systems. For example, a semantic-level query searching for images with green grass can be represented by a primitive image query which searches images with green colour in the bottom part. Since present CBIR systems cannot extract images' semantic features effectively, they cannot satisfy most semantic-level image queries.

2.2 Visual Features used in CBIR Systems

In a CBIR system, different visual features of images are automatically extracted and stored for any future retrieval process. Searching the whole image database is based on searching these visual features – metadata of real images. Colours, shapes, and textures are the most widely used in most CBIR systems.

2.2.1 Colour Features

Colour feature is one of the most widely used visual features in image retrieval since colour is immediately perceived by human beings [2] when looking at an image. Therefore, Colour-based Image Retrieval is the most popular CBIR technique. Using colour features in CBIR requires taking many factors into consideration: colour model selection, colour feature representation, and the metric to compute the distance between colour features.

Colour Models

The purpose of a colour model is to facilitate the specification of colours in a standard way [12]. In other words, a colour model is the quantitative way to represent colours that human beings perceive. Colour models used today can be classified into two categories: hardware-oriented and user-oriented [2, 12]. Hardware-oriented colour models are used for most colour devices. For instance, the RGB (red, green, blue) colour model is used for colour monitors and cameras; the CMY (cyan, magenta, yellow) colour model is used for colour printers; and the YIQ colour model is used for colour TV broadcast. User-oriented colour models including HLS, HCV, HSV, MTM and CIE-LUV, are based on the three human perceptions of colours, i.e., hue, saturation, and brightness. The selection of colour models determines the way to represent colour content of images as well as consecutive colour feature representations and the selection of image retrieval techniques. Therefore, a colour model is very important in a CBIR system. In order to understand many characteristics of colour features used in CBIR systems, some commonly used colour models are discussed next.

The RGB colour model is the most commonly used colour model. In the RGB colour model, a colour is represented by a combination of three primary colours: red, green, blue. The RGB colour model actually can be represented by a colour cube, as shown by Figure 2.1. As we can see, red, green, and blue colours are at three corners of the cube while cyan, magenta, and yellow are at three different

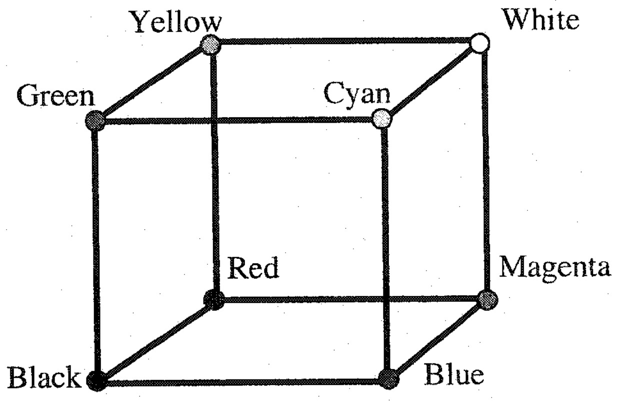


Figure 2.1: RGB colour model

corners. The values for red, green and blue increase respectively from the origin point along the axis on which they reside. The origin point represents colour BLACK, while the point furthest from the origin point represents the colour WHITE. The line connecting the BLACK colour point and the WHITE colour point represents all colours in grayscale. Gray colours on this line also change from pure black to pure white. All the other colours are represented as a point within the cube. The value for a specific colour is defined by the co-ordinate of the point in the cube along the red, green, and blue dimensions. Many kinds of image formats such as JPGs and GIFs store and show colours in the RGB colour model.

The CMY colour model is similar to the RGB colour model except that, in its colour model cube, the origin point is WHITE and the furthest point opposite to the origin point is the colour BLACK as shown in Figure 2.2. The CMY colour model is used mainly in colour printing. When a colour image displayed by a colour monitor is going to be printed by a colour printer, conversion from the RGB colour model to the CMY colour model is performed. The CMY colour can be obtained by converting from RGB colour as shown by Equation 2.1:

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.1}$$

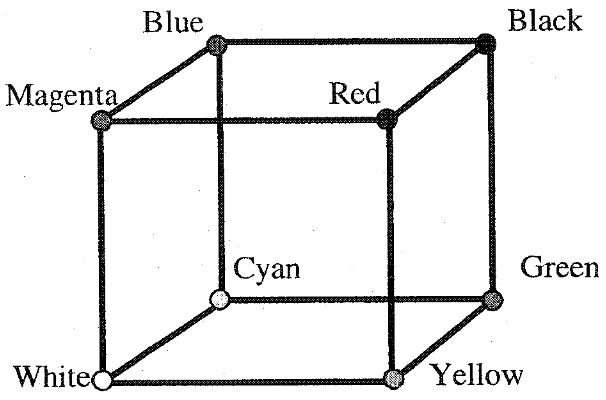


Figure 2.2: CMY colour model

There are many other colour models such as HSV, L$^*u^*v^*$, and YIQ. Among these, the RGB colour model has been the most widely used. Since many image formats, such as JPGs and GIFs, store and show colours in the RGB colour model, most CBIR techniques are based on this colour model. In addition, no evidence shows that other colour models can represent image visual features better than the RGB colour model obviously. Therefore, our thesis also uses techniques based on the RGB colour model. The discussion about colour features, as well as CBIR techniques, in the following sections will assume that the RGB colour model is used, except when specifically specified.

Colour Quantization

Prior to any processing being performed on a colour image, colour quantization is a very important step, due to the large number of different colours in the image. Under the RGB colour model, there are 256 different colour-levels (0-255) for each primary colour: red, green, and blue. That means, in a full-colour image under the RGB colour model, there are $16,777,216$ (256x256x256) possible colours, in total. If we operate with this large colour set, storage and processing will both be non-trivial. Actually, according to human perception, the difference between two adjacent colours in that large colour set is negligible. Therefore, keeping such a big colour set is neither practical nor necessary. Colour

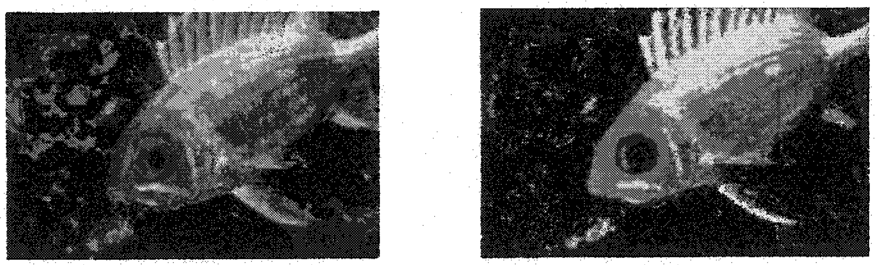


Figure 2.3: A colour image and its quantized image with only 4 colours

quantization is the procedure used to reduce possible colours to a small number. By using different quantization approaches, such as combining adjacent colours within a pre-defined range into one single colour, the large colour set can be reduced to a small number of possible colours. For example, an image can be quantized from true colour with 16777216 possible colours, to only 64 possible colours so that any needed processing on it would be easier. Figure 2.3 shows an example of an image quantized to only 4 colours. How we quantize images in our thesis is discussed in Chapter 3.

Colour Histogram and Normalization

Colour features of images are extracted and represented by some colour feature representations. There are many colour feature representation schemes, such as Colour Moments proposed by Stricker and Orengo in [34] and Colour Sets proposed by Smith and Chang in [31]. Among these, the colour histogram is the most commonly used representation of an image's colour feature, which represents the colour distribution of an image. The colour histogram can be represented as three independent colour distributions, in each primary colour: red, green, and blue. More frequently, it is represented as one distribution over the three primaries, obtained by counting how many pixels in an image belong to each different colour. Figure 2.4 shows an example image and its colour histogram.

The colour histogram for a given colour image S can be defined as

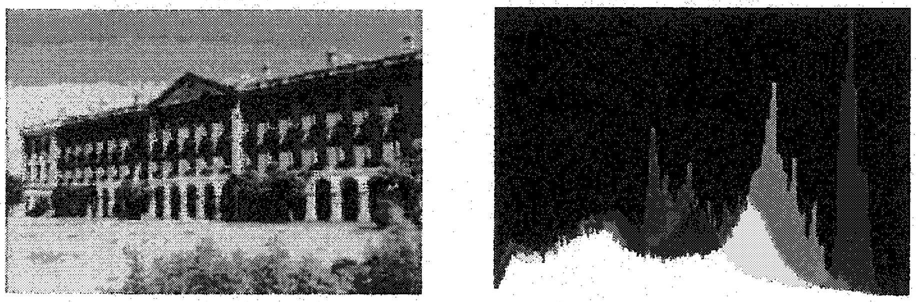


Figure 2.4: A colour image and its colour histogram

Colour	Number of pixels
(0, 0, 0)	234
(0, 0, 1)	23
(0, 0, 2)	478
.	.
.	.
.	.
(3, 3, 3)	3429

Table 2.1: Colour histogram for image A

$H^S = \{H_1^S, H_2^S, H_3^S, ...H_i^S, ...H_n^S\}$

where n is the number of quantization colours and H_i^S is the number of pixels belonging to the ith colour. For example, the colour distribution of the colour image in Figure 2.4, quantized to 64 colours, is shown in Table 2.1. According to the numbers of pixels belonging to each of the 64 different colours, the colour histogram of this image is: $H^A = \{234, 23, 478, ..., 3429\}$

The use of actual pixel numbers to denote the colour distribution of an image is easy to understand and manipulate. However, there is a problem when using this method to calculate the colour histograms for images of different sizes. Images of different sizes have different numbers of pixels. That means, for two images of different sizes, their colour histograms are not the same even when

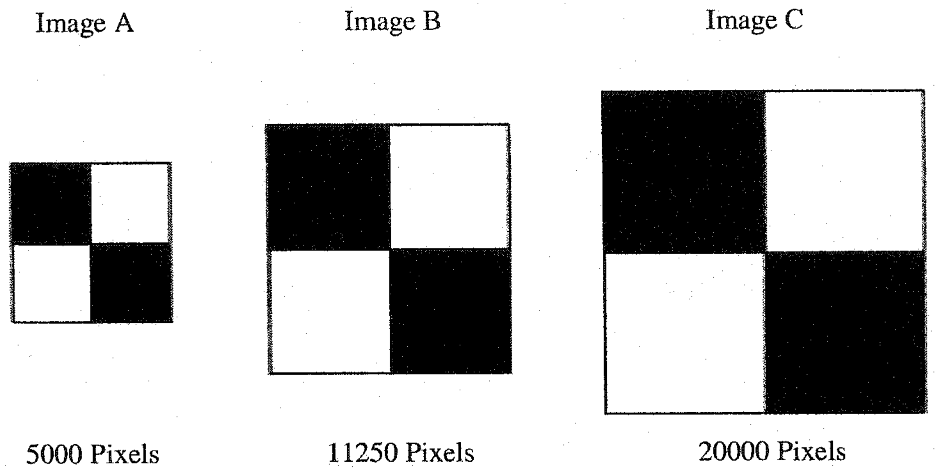


Figure 2.5: Three different size images with the same colour distribution

they have the same colour distributions. For example, suppose three images of different sizes are all quantized to 2 colours – black and white – as shown in Figure 2.5. The colour histograms for these three images are:

$$H^A = \{2500, 2500\}$$
$$H^B = \{5625, 5625\}$$
$$H^C = \{10000, 10000\}$$

As we can see, even though these three images have the same colour distributions, their colour histograms are different from each other, due to their different sizes. In order to make all colour histograms the same for images with the same colour distributions, despite their sizes, histogram normalization is necessary. Instead of using the actual number of pixels of each colour, percentage of dividing the number by the total image pixels is used in normalized colour histogram. In this way, as long as the colour distribution of images are the same, their colour histograms will be the same independent of their sizes. Normalized colour histogram for images in Figure 2.5 are :

$$H^A = \{50\%, 50\%\}$$
$$H^B = \{50\%, 50\%\}$$
$$H^C = \{50\%, 50\%\}$$

As we can see, normalized histograms for these three images are the same even when their sizes are different. In most CBIR systems, normalized colour histograms are used to represent the colour distribution of images.

2.2.2 Shape features

Shape information of objects in images is also a very important image visual feature. Searching image databases using shape-based techniques is very common in CBIR systems. Usually in such a CBIR system, shape features of all images in the database are extracted and indexed, as well as query images. The system then searches the database to find images with a similar shape features to the query image. Segmentation is the most important step during image shape feature extraction and includes many procedures such as noise removing and edge detection. Figure 2.6 is an example of shape feature extraction. As we can see, since objects in this image have clear and clean boundaries, the shape information is extracted accurately. There are some criteria to build a good Shape-based Image Retrieval system.

- Shape features extracted should be invariant to image rotation, scale and translation.
- Shape features should be able to be extracted from images easily and correctly without being affected by noise.
- The similarity algorithm to compute the distance between two shapes

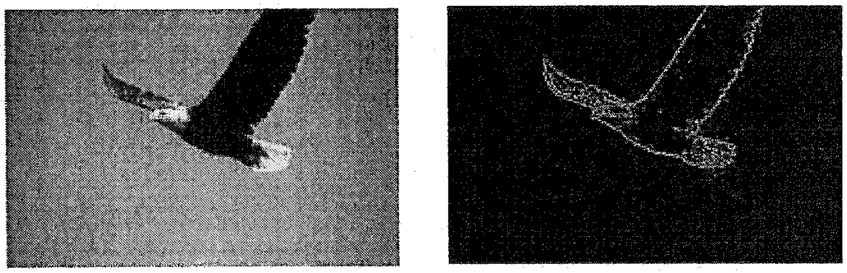


Figure 2.6: Shape extraction for an image with clear edges

should be similar to human perception. This means, images with more visual similar shapes of objects should have smaller distance between each other.

Shape-based Image Retrieval techniques can be categorized into boundary-based and region-based, according to shape feature representations. Boundary-based shape feature representation uses the outer edges of objects in an image, while region-based feature representation uses the entire shape region. Fourier Descriptor and Moment Invariants are the most famous techniques for these two categories, respectively. Fourier Descriptor [25] uses Fourier transformed edges as shape features. Moment Invariants [38] uses region-based moments as shape features which are invariant to transformations [24].

Some recent works, such as Mehtre's experimentations [18], show that using the combination of these two categories of shape feature representations outperforms using any single one. Also, with the emergence of 3D images nowadays, Shape-based Image Retrieval requires more research on capturing shape features of 3D objects correctly and retrieving similar ones from databases.

There is one main problem using shape features in CBIR: it can only deal with images with clear and clean boundaries. Accurate extraction of shape features of images with complex contours and significant noise is still being

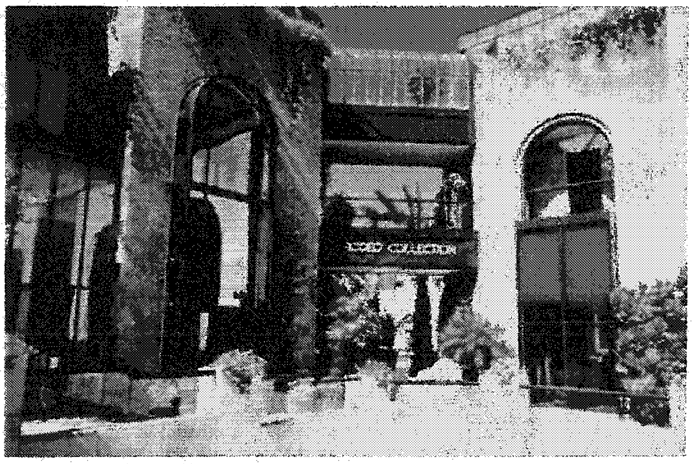


Figure 2.7: An image with complex objects

studied. As we can see from Figure 2.7, since contours of objects in the image are very complex and surrounded by many small objects which can be regarded as noise, extracting edges for that image is very difficult.

2.2.3 Texture features

"Texture features of images refers to the visual patterns that have properties of homogeneity that do not result from the presence of only a single colour or intensity" [29]. An image's texture content provides information of image properties such as smoothness, coarseness, and regularity which is useful in a CBIR system. Figure 2.8[1] shows examples of different image textures.

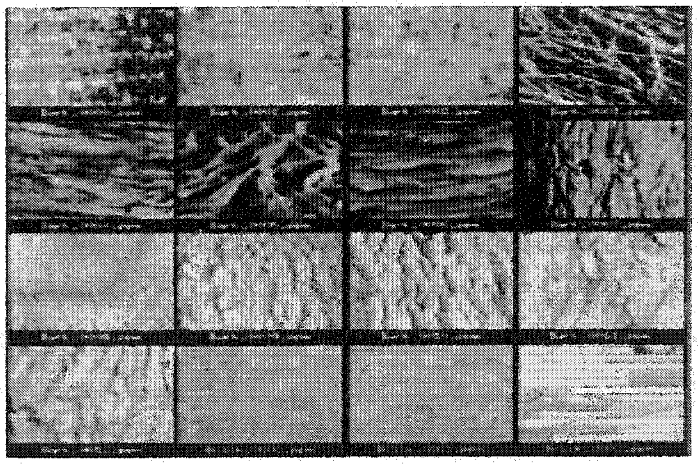


Figure 2.8: Different image textures

The general methodology of Texture-based Image Retrieval is described below. Images in a database, as well as query images, are first segmented into regions of same textures, and then a similarity measure is used to compute the distance between pairs of such regions. Finally, the distance between two images is the sum of distances between all pairs of regions with the same texture.

There are two main problems in Texture-based Image Retrieval [23].

[1]http://www-white.media.mit.edu/vismod/imagery/VisionTexture/photobook.gif

- Texture segmentation is difficult, and the notion of "same" texture is not well defined.
- Image retrieval based on comparing texture segments is usually sensitive to over-segmentation and under-segmentation.

Since there are some problems using only an image's textures as the visual feature to process image retrieval, in a practical CBIR system, texture features are always used in combination with other visual features, such as shapes and colours. The texture feature is especially useful in distinguishing between images with similar colour distributions.

2.2.4 Other features

Besides the image visual features we discussed, other image features exist and have been used. The spatial relationship between objects within an image carries information that a CBIR system needs to retrieve images similar to the query image. The spatial relationship may be classified into directional and topological relations [7]. The directional relationship denotes the corresponding position among objects within an image – such as east, west, south, north. On the other hand, the topological relation deals with the set-operational relations among objects. Under topological relations, objects can be disjoint, contains, inside, meet, overlap, equal, cover, and be covered-by. Much work has been done in the CBIR area based on spatial relationships. For example, 2D (Dimensional)-string based scheme [16] and the 9DLT (Direction Lower Triangular) based scheme [3].

Colour-spatial image retrieval technique is attracting increased research interest. This technique combines both colours and spatial features of images in order to facilitate image retrieval, and overcomes some drawbacks of using only one of them. Approaches such as partition-based scheme, signature-based algorithm and cluster-based methods have been studied [36].

As we discussed, several visual features can be selected for use in a CBIR system. However, in our work, we are emphasized on the use of colour features of images for the following reasons:

- Colour is the most straightforward visual feature of images and is perceived by human beings immediately.
- Although shape and texture features can also be used in CBIR systems, they both have some limitations which are not involved in the use of colour features. For example, shape-based image retrieval can only deal with images with clear and clean boundaries while texture-based image retrieval has difficulty in segmentation for getting images' textures.
- The colour feature of images can easily be extracted and manipulated in comparison with other features.
- Colour feature is still the most widely used visual feature in current CBIR systems [2].

Some colour-based image retrieval techniques are presented in the following sections.

2.3 Colour-based Image Retrieval Techniques

As mentioned, colour features are the most widely used in current CBIR systems. Many of these systems are based on image colour histograms, in which distance between images is measured by calculating the distance between colour histograms. Different histogram distance metrics, as well as some Colour-based Image Retrieval techniques, are discussed in this section.

2.3.1 Histogram Distance Metrics

There are a number of metrics for calculating the distance between two images' histograms. As an example, to explain different histogram distance metrics, we assume two images, A and B, are both quantized to 4 colours. Their normalized colour histograms are shown below:

$$H^A = \{20\%, 30\%, 10\%, 40\%\}$$
$$H^B = \{10\%, 10\%, 50\%, 30\%\}$$

The simplest way to calculate the distance between the two colour histograms is L1 distance. It calculates the absolute value of the difference between the same colours of two histograms and sums all of them as the total distance. As Equation 2.2 shows:

$$d(A,B) = \sum_{j=1}^{n} |H_j^A - H_j^B| \tag{2.2}$$

H_j^A and H_j^B are the normalized values in the colour histogram of colour j for image A and image B respectively. n is the total number of colours. So, for our example, the distance between two images, A and B, is:

$$d(A,B) = |0.2 - 0.1| + |0.3 - 0.1| + |0.1 - 0.5| + |0.4 - 0.3| = 0.8$$

The distance between two histograms can also be calculated using L2 distance as Equation 2.3 shows:

$$d(A,B) = \sqrt{\sum_{j=1}^{n} \left(H_j^A - H_j^B\right)^2} \tag{2.3}$$

H_j^A, H_j^B, j, and n denote the same meaning as in Equation 2.2. For our example, the distance between the two colour histograms using this metric is:

$$d(A,B) = \sqrt{(0.2 - 0.1)^2 + (0.3 - 0.1)^2 + (0.1 - 0.5)^2 + (0.4 - 0.3)^2} = 0.47$$

Using histogram intersection to calculate the distance between two colour histograms is another colour histogram distance metric proposed by Swain and Ballard in [35]. Equation 2.4 shows the histogram intersection distance metric between two colour histograms.

$$d(A,B) = 1 - \sum_{j=1}^{n} min(H_j^A, H_j^B) \tag{2.4}$$

Then, for our example, the distance using histogram intersection between two histograms is:

$$d(A,B) = 1 - (0.1 + 0.1 + 0.1 + 0.3) = 0.4$$

There are also many other histogram distance metrics. Swain and Ballard also proposed an alternative to the histogram intersection distance metric to improve retrieval effectiveness in large image databases, called incremental intersection [35]. It only calculates the distance between histograms using the colours that have the most pixels in query and database images. Niblack et al. have used a distance measure called weighted Euclidean distance in the QBIC system to evaluate the similarity of colour histograms [21]. In [11], Flickner et al. have proposed a simpler low-dimensional distance measure called average colour distance. Using this distance metric to calculate the distance between colour histograms can save significant time. However, L1-based and L2-based histogram distance metrics are still used the most widely in CBIR systems.

2.3.2 Global Colour Histogram (GCH) and Local Colour Histogram (LCH)

The GCH approach is the most popular Colour-based Image Retrieval technique and is often used as a benchmark for other techniques. In GCH, an image is represented by a single colour histogram to denote the whole content of the image. Colour image indexing and retrieval are both based on this single colour histogram. The distance between two images is calculated using one of the colour histogram distance metrics discussed earlier based on their GCHs.

Now let us give an example to show how GCH works in Colour-based Image Retrieval. Suppose image A and image B are the only two images in an image database. As shown in Figure 2.10, they are both quantized to 2 colours: black and white. As we can see, the GCHs for these two images are:

$$H^A = \{25\%, 75\%\}$$

$$H^B = \{50\%, 50\%\}$$

The query image's GCH as shown in Figure 2.9 is:

$$H^{QUERY} = \{75\%, 25\%\}$$

After we have the GCHs for all images in the image database, as well as the query image, different histogram distance metrics can then be used. Suppose we

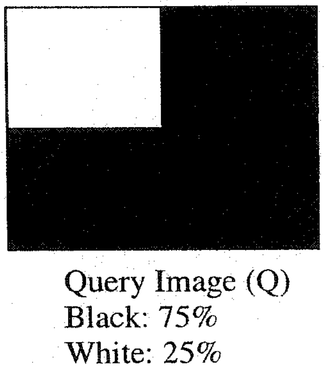


Figure 2.9: Sample query image (Q)

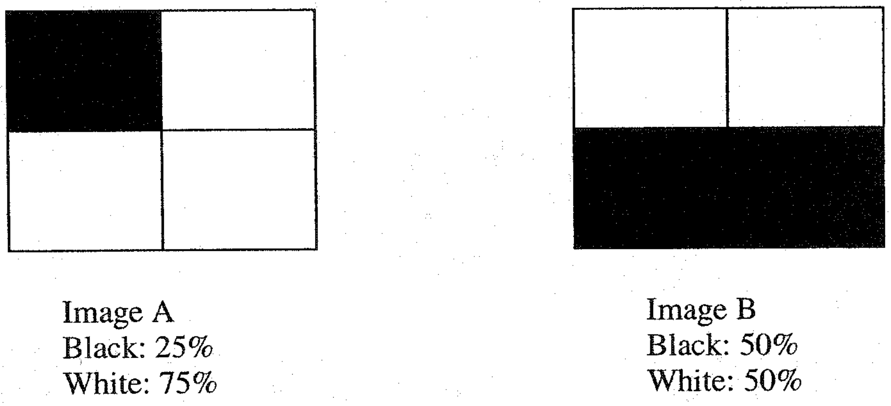


Figure 2.10: A sample image database with only two images, A and B

use the L1-based histogram distance metric. The distance between the query image and image A is:

$$d(Q, A) = |0.25 - 0.75| + |0.75 - 0.25| = 1$$

As well, the distance between the query image and image B is:

$$d(Q, B) = |0.5 - 0.75| + |0.5 - 0.25| = 0.5$$

Image B has a smaller distance to the query image than image A and, hence, Image B is supposed to be the most similar image in the database to the query image. Other histogram distance metrics such as L2-distance and histogram intersection can also be used in GCH image retrieval.

The idea behind GCH, as well as the implementation of it, is very simple. The global colour distribution of an image is represented by its colour histogram. In some cases, the GCH approach can give back acceptable query results. However, GCH performs poorly in many cases, since it does not take the localization information of colours into consideration. Two images which have a very small distance based on the GCH metric can have totally different visual appearances. LCH [8] is proposed as an alternative to overcome the shortcomings of GCH. Instead of considering only the global colour distribution of an image, information on local colour distributions is also extracted from the image. In LCH, an image is divided into many tiles, and GCHs for each of these tiles are extracted. Distance between two GCHs for each pair of tiles in the same location of two images is calculated. All these distances are summed to be the distance between two images. By combining the spatial features and colour features of an image together, LCH performs better then GCH in most cases in CBIR systems. However, the storage requirements and the processing time both increase. LCH is the essential CBIR technique we use in our thesis and will be discussed in detail in Chapter 3.

2.3.3 Other Techniques

There are many other Colour-based Image Retrieval techniques based on colour histograms, colour moments, and colour sets. Most of them attempt to combine colour information with other image features, such as spatial information, to improve image retrieval results.

Miller and Pass proposed Colour Coherence Vectors (CCV) in [19]. As colour histograms cannot capture the spatial information of an image, the CCV approach is a histogram-based scheme incorporating an image's spatial information in order to calculate the distance between two images. A colour's coherence is defined as the "degree to which pixels of that colour are members of large similarly coloured regions". These significant regions are referred to as coherent regions. Based on whether the pixels belong to a part of some sizable contiguous regions, the CCV approach classifies all pixels of an image into coherent and incoherent pixels. A colour coherence vector represents coherent and incoherent pixels for each different colour in an image. When calculating the distance between two images, coherent pixels in one image and incoherent pixels in another image will not match to each other even if they belong to the same colour. Experiments show that using CCV allows better distinction between images, which cannot be made using traditional colour histograms.

Based on the fact that humans intend to focus on large area of colours, rather than on small areas that are scattered around, a cluster-based image retrieval technique has been proposed by Tan, Ooi and Yee in [36]. Colour-spatial information of an image is represented by a set of single-coloured clusters, and used to facilitate image retrieval. Extracting clusters of an image consists of three phases. In the first phase, a set of colours which have the largest number of pixels in an image are selected. In the second phase, a cluster for each of these dominant colours is determined using a sequential 4-connected component algorithm. In phase three, these clusters are ranked according to their sizes. Then the top k clusters are selected as the dominant clusters of the image. Similarity between clusters of two images is calculated as the sum of overlapping pixel numbers for each pair of clusters of the same colour.

The Colour Shape Histograms (CSH) was proposed by Stehling, Nascimento and Falcao in [32]. This approach is based on the Local Colour Histogram and improves the colour histogram encoding of LCH. In LCH or GCH, the percentages of colours not presented in an image will be zero. In CSH, the colours not presented in an image will not be encoded into colour histograms at all. Experiments show that the number of actual different colours presented in an image is considerably fewer than the colours in the RGB colour model. In an image having N colours, there will be only N CSHs. This can reduce the storage of image colour histograms and increase the efficiency of image retrieval.

In [34], the authors proposed a technique for image colour indexing based on colour moments. In their approach, different from GCH, an image's colour content is represented by only their dominant features instead of storing the complete colour distributions. They characterize a colour representation of image only by the first three colour moments: colour average, colour variance, and colour skewness. The distance between two images is calculated as a weighted sum of the absolute differences between corresponding moments. Thus, the similarity of images is determined by their colour moments. Through using this colour representation, low space overhead is achieved.

Colour histogram sets are proposed in [5] by Colombo, Rizzi and Genovesi to represent the local colour properties of images. An image is segmented into a set of regions by dividing the image into small non-overlapping tiles, and then clustering them with a split and merge technique. Histogram intersection is then used to measure the degree of colour distribution homogeneity between pairs of image regions. This approach is suitable for images with regions having a colour distribution which significantly differs from the global colour distribution of the image. There are many other Colour-based Image Retrieval techniques, for an extensive survey refer to [2].

2.4 CBIR Systems

Since Content-based Image Retrieval has recently been an active research area, many different CBIR systems have been developed. In this section, we select some of them and describe their characteristics.

Query By Image Content (QBIC) [21] is the first and most famous CBIR system and was developed by IBM Almaden Research Center. It is available either in stand-alone form, or as part of other IBM products, such as the DB2 digital library. Visual features used in QBIC include colours, shapes, and textures. The colour features used are 3-dimensional vector in RGB, YIQ, L*ab* and Munsell colour model. The shape features consist of shape area, circularity, eccentricity, major axis orientation, and a set of algebraic moment invariants [11]. Coarseness, contrast, and directionality are texture features used in the system. In QBIC, different distance metrics such as histogram-based algorithms are used to calculate the distance between two images. QBIC supports queries based on example images, user-constructed sketches and drawings, selected colours, and texture patterns. A demo of QBIC can be seen at http://wwwqbic.almaden.ibm.com.

The Photobook system [22] was developed by Pentland, Picard and Sclaroff from the Media Laboratory of MIT, in 1995. It is a set of interactive tools for browsing and searching images in an image database. The key idea behind this suite of tools is semantic-preserving image compression, which reduces images to a small set of perceptually significant coefficients. Photobook has been developed based on three different visual features: appearances, 2-dimensional shapes, and texture properties. All these representations can be combined to provide users with a more sophisticated and efficient utility to browse and search images. Many distance metrics such as Euclidean, mahalanobis, divergence, vector space angle, and Fourier peak are used in the system. Users can also define their own distance metrics in the latest version. A demo of Photobook can be found at: http://vismod.www.media.mit.edu/vismod/demos/photobook/index.html.

The VIRAGE Image Engine is another CBIR system, developed by Virage Technologies, Inc. It uses colours, shapes, and textures as visual features of images. These visual features can be used separately or combined together to provide a more specific image searching. One of the advantages of using the VIRAGE system is that users can adjust the weights associated with the atomic features according to their own emphasis [24]. A VIRAGE demo can be found at: http://www.virage.com/cgi-bin/query-e.

The image searching engine, NETRA, was developed by University of California, Santa Barbara [17]. Images are segmented into regions with the same colour. For each of these regions, the colour, texture, shape and spatial features are extracted as visual features of images. These features can then be combined to search and retrieve similar regions from the database. In other words, this representation allows the user to compose interesting queries such as "retrieve all images that contain regions that have the colour of object A, texture of object B, shape of object C, and lie in the upper of the image" [17]. The online demo of Netra can be found at: http://vivaldi.ece.ucsb.edu/Netra/.

There are many other CBIR systems, such as Amore, developed by C&C Research Laboratories NEC [20], CANDID, developed by Los Alamos National Laboratory [15], ImageRover, developed by Boston Univ. [26], and Columbia University's VisualSEEK&WebSEEK [28, 30].

Chapter 3

Mosaicture Methodology

3.1 Image Mosaic and Related Work

An image has its own visual content, such as a bird flying in the sky or a little girl swimming in the sea. If several of these images are put together, what will the visual content of that large image be? Obviously, if many images are put together randomly, they will not suggest a meaningful content as a whole. Selecting images and putting them together, according to some methodology, in order to make the large image have its own meaningful visual content is an

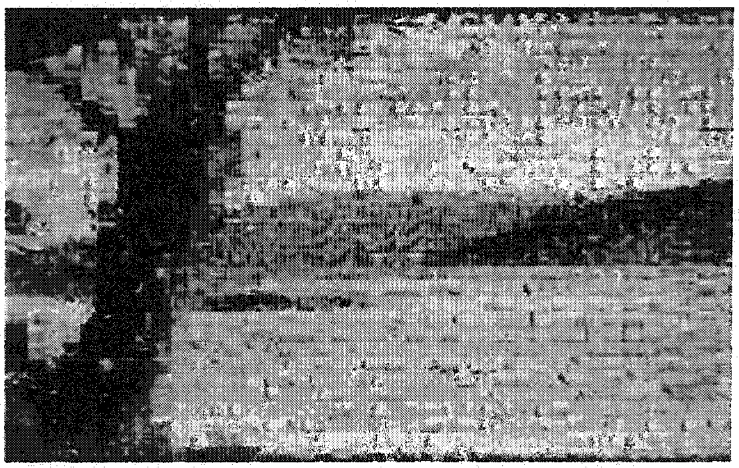


Figure 3.1: An image mosaic of landscape by Mosaic Magic [1]

interesting topic which has been studied for a long time. These "big images" are called image mosaics or image montages. Figure 3.1 shows such an image mosaic. The history of image mosaics as well as some existing image mosaic generating methodologies and products is discussed in the following sections.

3.1.1 Image Mosaics – History and Present

Building an image by combining many other images has a long history. Dating back to the 16th century, the painter Giuseppe Arcimboldo manually put pictures of vegetables, fruit and flowers together to build a whole image. American painter, Chuck Close, generate many portrait using greyscale blocks. Figure 3.2 shows one of his self-portraits. In November 1973, an article, "The Recognition of Faces", written by Leon Harmon, and which used a portrait of Lincoln as an example of recognizing human faces, was published in Scientific American. Different from common images, Lincoln's face in this portrait is suggested by combining together many grayscale blocks. In 1976, instead of using just grayscale blocks, Salvador Dali used small colour images to build an image mo-

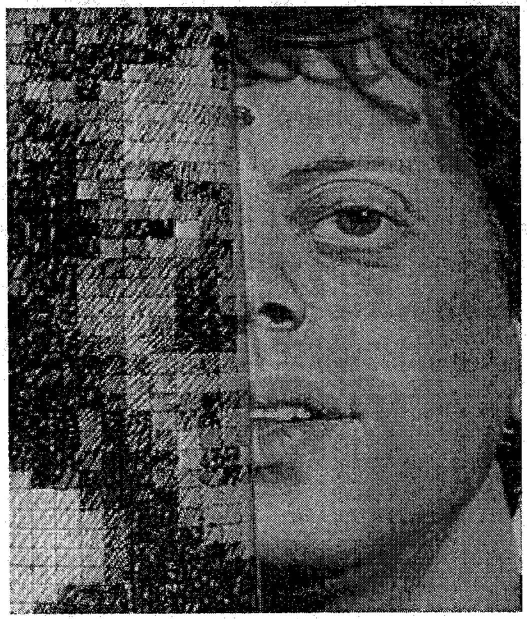


Figure 3.2: Chunk Close's self-portrait

saic of the same portrait. In Dali's work, he put his wife's picture in the center of Lincoln's face with some other small images of Lincoln himself scattered around. This mosaic image is likely one of the most famous art works in this area and is shown in Figure 1.1 in Chapter 1.

With the advent of advanced computer technologies, both artists and computer scientists are trying to build image mosaics with the help of computers.

A famous image mosaic generating system using computers is Robert Silver's "Photomosaic" [27]. This system generates an image mosaic by putting many images together according to some algorithms. Different options are available for users to control the generating process. Robert Silvers patented his technique of building image mosaics in 1996. His company, Runaway Technology, now generates image mosaics for many organizations, companies, and academic institutions. Many of his image mosaic products can be seen on the Internet at http://www.photomosaic.com.

Obviously, image mosaics have their value both in arts and entertainment. In addition, image mosaics can also be used in the area of copyright protection. For instance, it is not safe to let just anybody have access to some original priceless masterpiece of art. Reproductions of these original art works can be produced in the format of image mosaics which can then be shown to people. People can still see the content of the art work, but they cannot produce fake works based on these previews, due to the insufficient information contained. Other applications of image mosaics, such as compression and complexity theory, are discussed in detail in [37].

3.1.2 General scheme of image mosaic generating systems

Typically, an image mosaic is generated as a reproduction of its original image. In other words, when an image called the original image is given, many images are put together to build an image mosaic which has a similar visual content to the original image. The typical scheme of generating image mosaics is described below.

In an image mosaic generating system, thousands of images, which will be used as building blocks to generate an image mosaic, are contained in an image database. All these database images are pre-processed by the system. During this step, visual features such as colours, shapes, and textures are extracted from database images and stored as metadata of real images. When an original image is given, it is divided into many blocks called tiles. Then for each tile, the same visual features are extracted and used to match the metadata of database images. According to some criteria, the image in the database with the most similar visual content is selected as the best matching image for that tile. Finally, all these best matching images for each tile are combined to build up an image mosaic reproduction of the original image. For the purpose of making the content of an image mosaic generating system understandable through this thesis, some terms we just used to describe the image mosaic generating process are explained below:

- Original image - The image based on which an image mosaic reproduction is generated.
- Target image - The final image mosaic reproduction of the original image.
- DB image - The source image in an image database which is used to build an image mosaic.
- Tile - When an original image is divided into many small blocks, each of these small blocks is called a tile.
- Best matching image - The image within an image database that is considered by the system having the most similar visual content to a tile.

3.1.3 Existing Methodologies and Products

Even though building image mosaics is an interesting topic, it still has not been studied widely. However, we describe some existing methodologies as well as introducing some commercial products in this section as the background of our research.

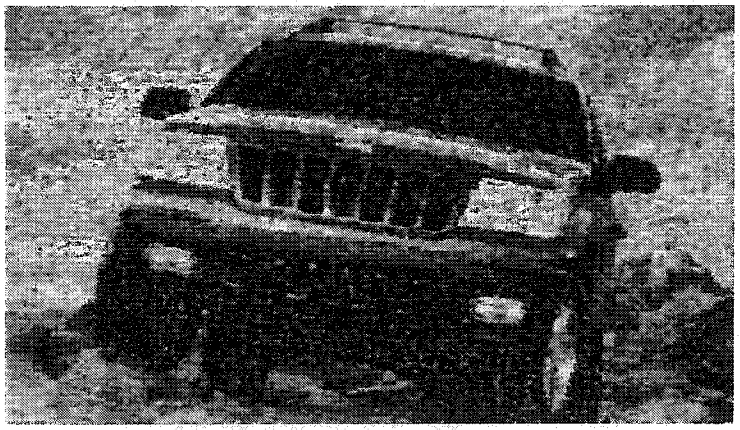


Figure 3.3: An image mosaic generated by Photomosaic [27]

As mentioned previously, Robert Silvers' patent system "Photomosaic" is probably the most famous image mosaic generating system. In Silvers' Photomosaic system, DB images can be both regular images and snapshots captured from other devices, such as a VHS video tape player. In addition, these images are cropped to be square in the same dimension. All these DB images are classified into many categories according to their semantic content such as animals, humans, and nature. Users of the system can select using DB images in a specific category to generate an image mosaic. Also, these DB images are stored in different resolutions. Images in low resolution are used to generate an image mosaic, while images in high resolution are used to print out a poster of the bigger image. When an original image is loaded into the system, it is divided into many tiles. For each tile, the system compares its visual content with all DB images and computes the distance between the tile and the DB images. Since the implementation of the system is protected by US patent policy, we do not know the details of the matching process. This matching process goes through each and every tile of the original image and finds the best matching image for each of them. Finally, all the best matching images are put together to generate the target image mosaic. The image mosaics generated by Robert Silvers' Photomosaic have a unique appearance, while keeping similar visual content to the original image. Figure 3.3 shows an example. However, in order to make

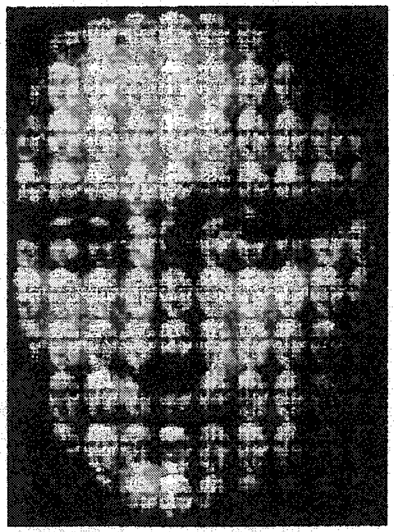


Figure 3.4: Generating an image mosaic by adjusting colours [10]

the image mosaic have a very similar appearance with the original image, some manual adjustment is needed [33].

Another process for creating an image mosaic is described in [10]. Different from Robert Silvers' Photomosaic, Finkelstein and Range pay more attention to colour adjusting, after putting many images together. Four steps make up the whole process: (1) choosing DB images, (2) choosing a tiling grid, (3) finding an arrangement for the DB images within the grid, and (4) correcting the colours of DB images to match the original image. When choosing DB images, the authors manually select images which can be recognized easily, such as political leaders, actors, and well known scenes. During the step of arranging DB images to form the image mosaic, the authors try many approaches – such as using the same DB image everywhere, randomly putting images together, and arranging images manually. The fourth step – adjusting colours of images – is what they are mostly concerned with. The authors make use of a colour correction rule, which adjusts a DB image in the target image mosaic to have the same average colours as the corresponding tile in the original image. The image mosaic generated using this approach has a very similar colour content to the original image. Figure 3.4 shows such an image mosaic generated by only adjusting the colours of DB images. However, since many steps in this approach are done manually, the processing time is a big problem. In addition, since this approach generates an image mosaic mainly by colour adjusting, the real colour content of DB images is lost.

Nicholas Tran also describes two algorithms for generating image mosaics [37]. In that paper, instead of a whole image mosaic generating system, only the procedure for finding the best matching image for an input tile is studied. Neither of these two algorithms tries to perform any shapes, texture, or colour analysis of images. They are simply based on the distance between the pixels of two images. The first algorithm calculates the distance between a DB image and a tile by summing all the absolute values of difference between two pixels at the same location in two images. The DB image with the smallest distance is selected as the best matching image. Instead of calculating the distance between two images based on pixels, the second algorithm calculates the distance based on rows of pixels. In this paper, the author also proposed a method to evaluate an image mosaic's quality. The quality of an image mosaic is determined by the similarity between the original image and itself. This similarity is measured as the distance the author is standing from the image mosaic and its original image so that he cannot tell the difference between them. The smaller this distance is, the better the quality of the image mosaic. The two algorithms proposed in Tran's paper are easy to understand and implement. However, both of them just use the very simple pixel-by-pixel-based algorithm to calculate the distance between an image and a tile, which do not take the global content into consideration. Also, their method of evaluating the quality of an image mosaic is purely based on human perception. Even though human eye can be considered as the best way to evaluate an image mosaic's quality, it is not practical to evaluate hundreds of image mosaics, one by one, using this method.

In addition to what we have discussed above, there are other commercial products for generating image mosaics. PhotoShop[1] from Adobe cannot build image mosaics automatically, but it is still possible to manually manipulate

[1]http://www.adobe.com/products/photoshop/main.html

images and combine them to form a large image. ArcSoft's PhotoMontage[2] is another commercial product which can build image mosaics automatically. It provides users many options to control the image mosaic generating process, such as the number of times a DB image can occur in an image mosaic. Users can also add their own images into the image database included in this software. Figure 3.5 shows an image mosaic generated by Photomontage. Mosaic Magic [1] is a free software for building image mosaics, which also permits users to add their own images to the existing database. In addition, it provides a colour adjusting function with which users can adjust the colour of an image mosaic to get a better result.

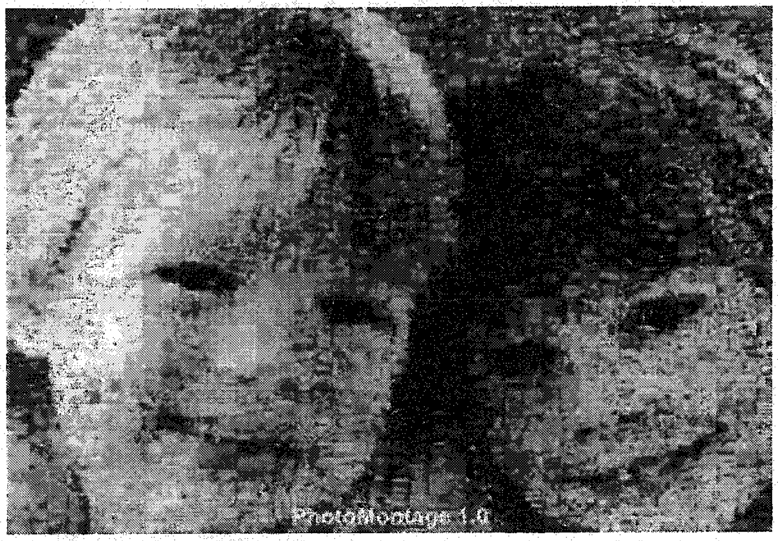


Figure 3.5: An image mosaic created by PhotoMontage1.0 [14]

[2]http://www.arcsoft.com/products/software/en/photomontage2000.html

3.2 Mosaicture – Our Image Mosaic Generating Methodology

3.2.1 Architecture of the system

Mosaicture, the image mosaic generating methodology that we propose, is implemented as an image mosaic generating system in our thesis. The whole system consists of two main stages, as shown in Figure 3.6:

- Image Database Pre-processing Stage.
- Image Mosaic Generating Stage.

For the purpose of finding the best matching image for a tile when generating an image mosaic, the visual feature of the tile as well as the DB images must be extracted and compared. In the first stage, an image database containing thousands of images is pre-processed. Colour features of all DB images are extracted and represented by binary signatures, as described in [4]. These signatures are stored as the metadata of real DB images. There are many parameters used during this stage to control the pre-processing process – such as the quantization colours of DB images and the size of the image database. In the second stage, an original image is loaded into the system which is then divided into thousands of tiles. Then, for each tile, the system retrieves the best matching image within the image database, and replaces the tile with the image. This procedure is actually a CBIR procedure. When computing the distance between a tile and DB images, a binary signature based distance metric is used with the combination of the scheme we proposed to select the best matching image. Finally, all the best matching images are put together to make up a target image mosaic. In the following sections, our image mosaic generating system is discussed in detail with respect to these two stages.

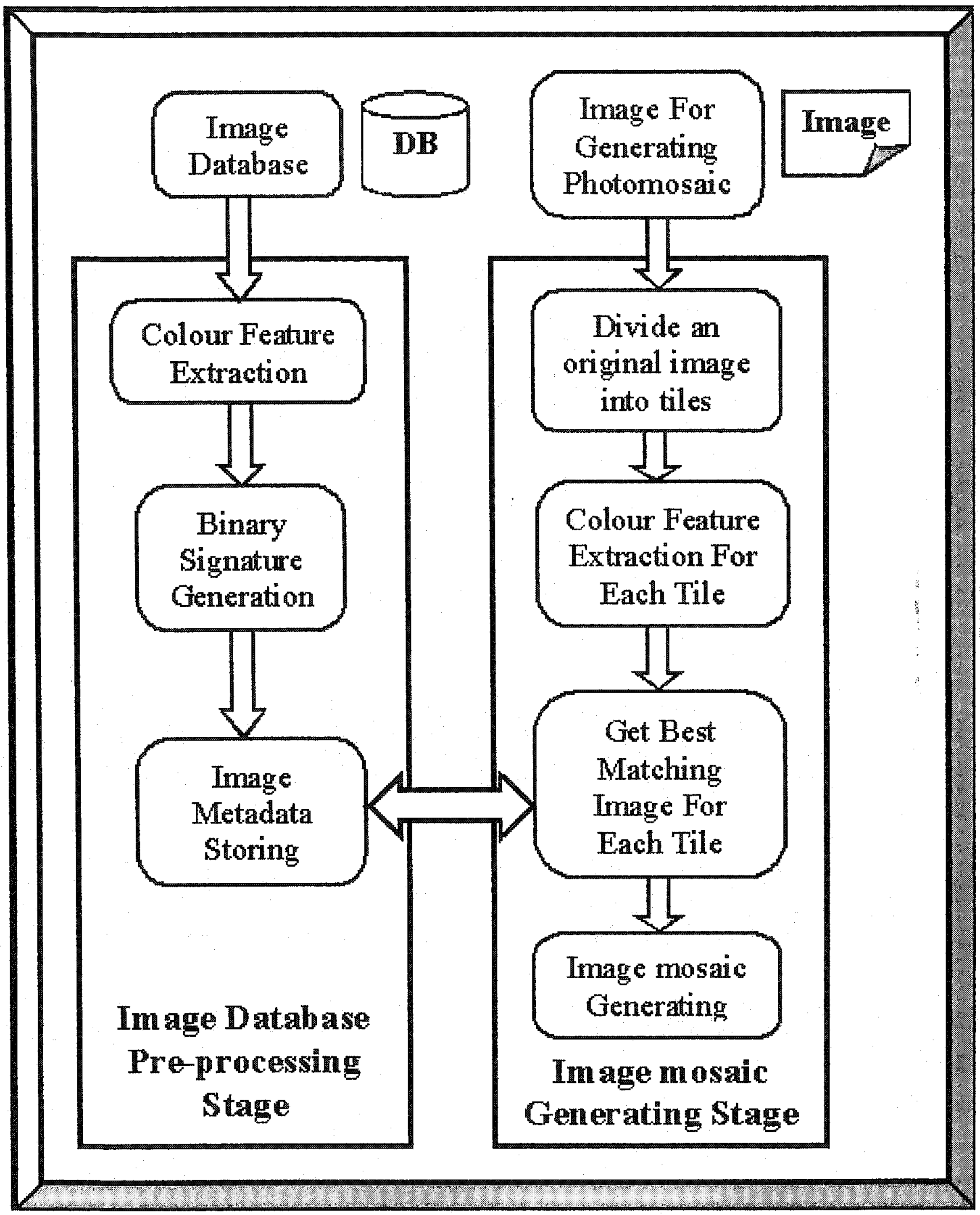


Image Database
DB
Image For Generating Photomosaic
Image
Colour Feature Extraction
Binary Signature Generation
Image Metadata Storing
Image Database Pre-processing Stage
Divide an original image into tiles
Colour Feature Extraction For Each Tile
Get Best Matching Image For Each Tile
Image mosaic Generating
Image mosaic Generating Stage

Figure 3.6: The system architecture of Mosaicture

3.2.2 The image database pre-processing stage

In our system, the image database pre-processing stage includes many procedures such as extracting visual features from DB images and representing visual features with abstract representations. All these procedures, as well as some parameters used, are discussed in detail in the following sections.

Selecting DB Images

An image database contains all the DB images for generating image mosaics. First, we select our DB images. There are some factors we need to take into consideration, such as the format of images and the number of images in the database. There are many different image formats available such as JPGs, GIFs, and BMPs. Among these image formats, JPG images are the most widely used in digital archives, as well as on the Internet. In comparison with other image formats, JPG images are smaller in size, while maintaining good quality at the same time. In addition, JPG images use the RGB colour model, which is simple and easy to manipulate. Thus, we select images in JPG format to make up our image database. We also need to decide how many DB images we should have in the database; this is actually a parameter in our system. This parameter affects the processing time for generating an image mosaic, as well as the target image mosaic's quality. Typically, the more DB images we have, the better the target image mosaic will be, as well as the more processing time is spent to generate such an image mosaic. Analysis of the experimental results of this parameter will be discussed in detail in Chapter 4.

Image Quantization

Since colour features are the most straightforward visual features of images, we use CBIR techniques based on colour features when searching the best matching image for the tiles of an original image. Before we can extract colour features from DB images, colour quantization is a necessary step. As mentioned in Chapter 2, colour quantization is used to reduce the number of colours of a full-colour image. There are two main colour quantization approaches: Uniform Colour Quantization and Non-uniform Colour Quantization. In the first

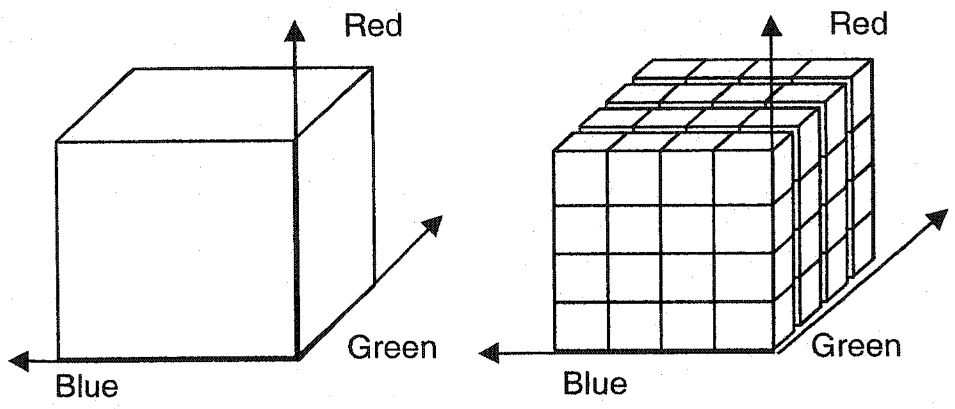


Figure 3.7: Quantizing the RGB colour model to 64 (4^3) colours

approach, all images are quantized to the same set of colours. These colours are produced by combining many different colours within a pre-defined range into a single colour. In the non-uniform quantization approach, more colours might be used in the colour levels that are perception-sensitive. Under this approach, different images might be quantized to different sets of colours due to their different colour contents.

In our work, we need to compare an input tile of an original image with all DB images in order to find its best matching image. We therefore select using the uniform quantization approach to quantize all images to the same set of colours. As mentioned in Chapter 2, the RGB colour model can be viewed as a cube; any single point within this cube represents a colour. When we perform the uniform colour quantization, we divide this cube into many sub-cubes by separating each axis, Red, Green, and Blue into the same number of sub-divisions. Each of these sub-cubes represents a quantized colour. Figure 3.7 shows an example of producing 64 quantization colours. In this example, each axis is divided into 4 sub-divisions, thus the cube is divided into 64 (4^3) sub-cubes. Each of these 64 sub-cubes represents a single colour after quantization.

In our work, a colour (R, G, B) in a true-colour image is converted to the colour (r, g, b) in a quantized colour space according to Equation 3.1,

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} R * n/256 \\ G * n/256 \\ B * n/256 \end{pmatrix} \tag{3.1}$$

where n is the number of sub-divisions into which each of the three primary colours is divided. For example, a true colour (23, 242, 162) is converted to the colour $(23 * 4/256, 242 * 4/256, 162 * 4/256)$ in a 4^3 quantization colour space. In other words, the original colour is converted to (0, 3, 2).

There is no best number of quantization colours. Typically, the fewer colours used, the faster and more easily an image can be manipulated, but in the mean time, the more colour information of an image is lost. Actually, the number of quantization colours is an important parameter in our system. We select three different quantization colours: 8 (2^3), 27 (3^3), and 64 (4^3), as possible choices in order to test their effects on the process of generating an image mosaic.

Using LCH

After all DB images have been quantized, we can perform the process of colour feature extraction. Among all the colour feature representations we discussed in Chapter 2, GCH and LCH are probably the most widely used since they are easy to understand and simple to manipulate, in comparison with the others. In the GCH approach, an image's global colour distribution is extracted and used as the image's metadata. If users searching images within an image database only care about the global colour content of images, using GCH is a good choice. However, since GCH only takes the global colour distribution of images into consideration when comparing images, it could give back retrieval results conflicting with actual visual perceptions. This can be explained by the following example.

Image A, B, and C are all quantized to 3 colours: black, gray, and white as shown in Figure 3.8. The GCHs for these three images can be calculated based on how many black, gray, and white small blocks are contained in each image.

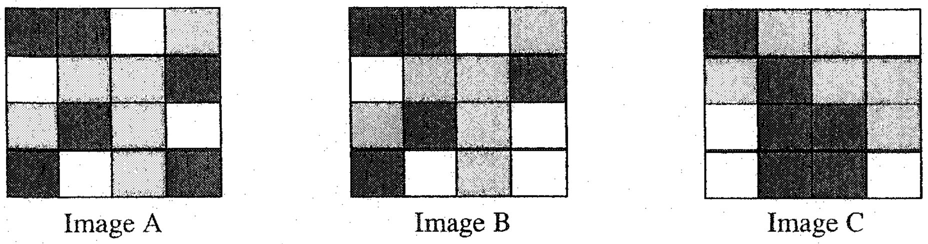


Figure 3.8: Three images with different colour distributions

Image	BLACK	GREY	WHITE
A	37.5%	37.5%	25%
B	31.25%	37.5%	31.25%
C	37.5%	37.5%	25%

Table 3.1: GCHs for Image A, Image B, and Image C

Table 3.1 shows their GCHs. Assume image A is the query image, and images B and C are DB images. Based on the L1 distance metric, the GCH distances between the query image and DB images are:

$$d(A,B) = |0.375 - 0.3125| + |0.375 - 0.375| + |0.25 - 0.3125| = 0.125$$
$$d(A,C) = |0.375 - 0.375| + |0.375 - 0.375| + |0.25 - 0.25| = 0$$

From the result of comparing the GCHs of images, image C is found to be more similar to the query image than image B. However, this is not true according to our perception of Figure 3.8, which shows that image B is more similar to the query image. This illustrates the shortcomings of using GCH, which does not store the spatial information of images.

In our work, when generating image mosaics, all the best matching images for every tile of an original image are put together. This kind of organization of DB images determines that we care not only about the global colour contents of an image but also about the image's colour localization. This can be explained by the following example.

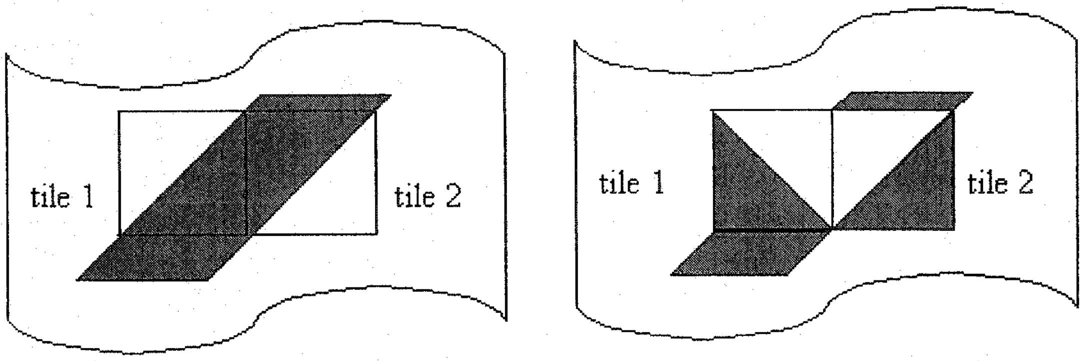


Figure 3.9: Using two tiles having the same GCH as the original image may result in an image mosaic with very different visual content

As shown in Figure 3.9, consider the left-hand image in the figure is part of an original image which has a black stripe in it. When the original image is divided into many tiles, two tiles, tile 1 and tile 2, are contained in this part. What we want here is to use a DB image containing black content in its southeast part to replace tile 1, and a DB image containing black content in the northwest part to replace tile 2. When searching such DB images, both colour and spatial information of images must be used to decide whether an image has the right colour content in the right place. If only the GCH is used, a DB image could be considered as the best matching image for a tile as long as it has the same global colour contents, no matter where they are. Then we may see a resulting image mosaic which is totally different from the original image as shown by the right-hand image in Figure 3.9, even though the best matching images selected have the exact same GCHs as their corresponding tiles. In order to avoid this problem, we choose to use the LCH instead.

The LCH is proposed as an improvement of the GCH by encoding images' local colour distribution as well. Typically, an image is divided into many blocks, and the GCH for each of these blocks is extracted. For example, an image divided into 16 blocks has 16 colour histograms. Each of these 16 colour histograms is the GCH for its corresponding block. The distance between two images is the sum of the distance between the GCHs of blocks in the same location in two images.

In most cases, using the LCH outperforms using the GCH due to the spatial information of images encoded. Using the three images in Figure 3.8 again as an example, assume each image is divided into 4 blocks to obtain LCHs, as shown in Figure 3.10. The LCHs for image A and image B are shown in Table 3.2. The distance between image A and image B, based on the L1 distance, is shown as Equation 3.2:

$$\begin{aligned} d(A,B) &= |0.5-0.5|+|0.25-0.25|+|0.25-0.25|+ \\ &\quad |0.25-0.25|+|0.5-0.5|+|0.25-0.25|+ \\ &\quad |0.5-0.5|+|0.25-0.25|+|0.25-0.25|+ \\ &\quad |0.25-0|+|0.5-0.5|+|0.25-0.5| \\ &= 0+0+0+0.5=0.5 \end{aligned} \tag{3.2}$$

Using the same approach, as shown in Figure 3.11 and Table 3.3, the distance between image A and image C is shown as Equation 3.3:

$$\begin{aligned} d(A,C) &= |0.5-0.5|+|0.25-0.5|+|0.25-0|+ \\ &\quad |0.25-0|+|0.5-0.75|+|0.25-0.25|+ \\ &\quad |0.5-0.5|+|0.25-0|+|0.25-0.5|+ \\ &\quad |0.25-0.5|+|0.5-0.25|+|0.25-0.25| \\ &= 0.5+0.5+0.5+0.5=2 \end{aligned} \tag{3.3}$$

According to the comparison result using LCH, image B is more similar to the query image A than is image C, which conforms with our visual perception. The LCH outperforms the GCH by taking images' local colour distribution into consideration when comparing two images. Under the LCH scheme, two images are considered the same only when they have the same colour content in the same location. This is also what we need after the discussion of the example shown in Figure 3.9.

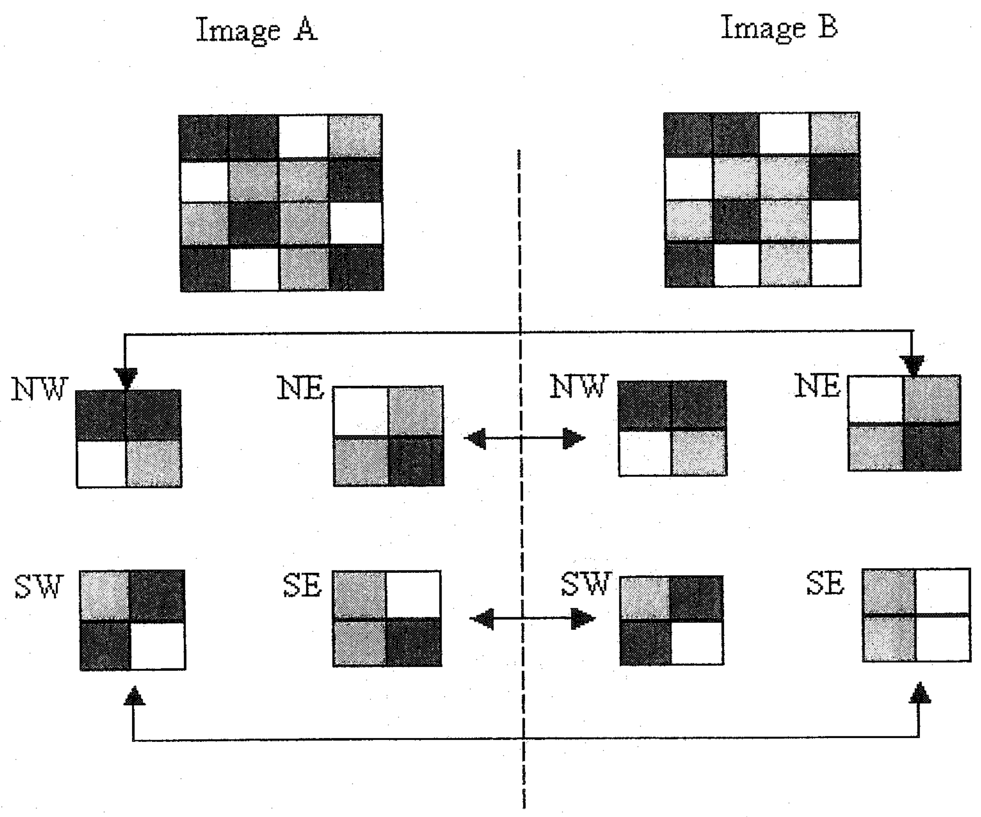


Figure 3.10: Comparing Image A and Image B using their LCHs

ImageA	BLACK	GREY	WHITE	ImageB	BLACK	GREY	WHITE
NW	50%	25%	25%	NW	50%	25%	25%
NE	25%	50%	25%	NE	25%	50%	25%
SW	50%	25%	25%	SW	50%	25%	25%
SE	25%	50%	25%	SE	0%	50%	50%

Table 3.2: LCHs for Image A and Image B

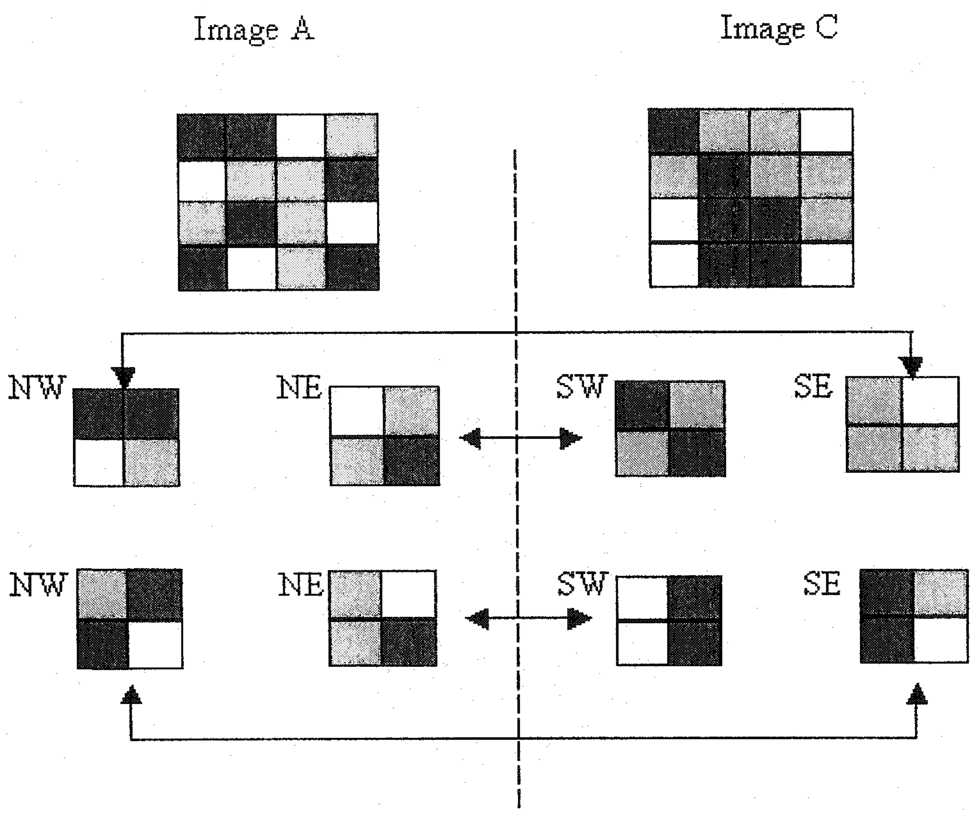
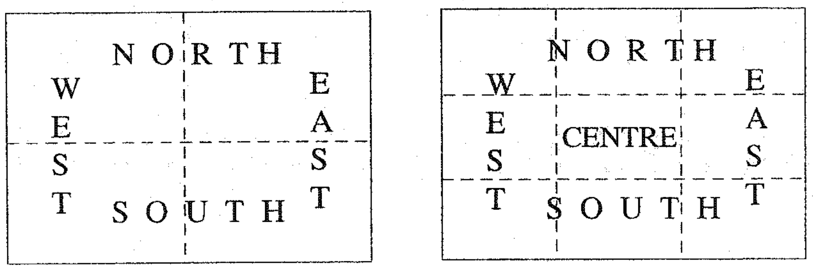


Figure 3.11: Comparing Image A and Image C using their LCHs

ImageA	BLACK	GREY	WHITE	ImageC	BLACK	GREY	WHITE
NW	50%	25%	25%	NW	50%	50%	0%
NE	25%	50%	25%	NE	0%	75%	25%
SW	50%	25%	25%	SW	50%	0%	50%
SE	25%	50%	25%	SE	50%	25%	25%

Table 3.3: LCHs for Image A and Image C

There are many different schemes to extract LCHs according to the way an image is divided into blocks. For example, when extracting LCHs, an image can be divided into blocks of the same or different sizes, and these blocks can be separated or overlapped. Different schemes have different advantages in different cases. Typically, the more blocks an image is divided into, the more local colour distribution information is encoded, as well the more storage and processing time required. In our work, DB images are all small, since hundreds of them are used to generate just one image mosaic. As well, after dividing an original image into many tiles, not much local colour distribution can be seen for each tile. Therefore, dividing the images in our system into a large number of blocks when extracting their LCHs is not necessary. In addition, in order to generate just one image mosaic, we have to find the best matching image for each and every tile of an original image. This is a very time-consuming process and does not allow us to store too much information for an image. So, we use dividing an image into 4 blocks to extract its LCHs. These 4 blocks are produced by dividing an image according to its physical layout: North, East, South, and West. They overlap with each other partially, as shown in Figure 3.12. Using 4 LCHs for an image speeds up the colour feature extraction process, as well as the future CBIR process, while keeping sufficient information about an image's local colour distribution. In addition, we try another approach – dividing an image into 5 blocks: North, East, West, South, and Central, as shown in Figure 3.12. As



Divide an image into 4 parts Divide an image into 5 parts

Figure 3.12: Dividing images into 4 and 5 parts to get LCHs

well as one more block divided than in the first approach, the central part of an image is emphasized, based on the fact that it typically contains the principal content of an image, such as the main object. Actually, the scheme used to divide an image so that LCH can be extracted is a parameter in our image mosaic generating system. Experimental results and analysis of this parameter are discussed in detail in Chapter 4.

Using Binary Signature

After the LCHs for all DB images have been extracted, in a typical CBIR system, these LCHs are then stored as the metadata of DB images. When a query image is presented by a user, the LCHs of this image are also extracted and used to match the metadata of DB images. Finally, the DB images with the smallest distance to the query image are retrieved back by the CBIR system as the query result. However, using colour histograms directly as described above is not practical due to our work's characteristic. Different from a typical CBIR system, which processes just one query image at a time, our system needs to process thousands of query images to generate just one image mosaic. In our system, an original image is divided into thousands of tiles, and each tile must be processed by the system to find the best matching image. This makes the whole image mosaic generating process a very time-consuming work. Due to the high computation cost and storage requirement using colour histograms, binary signature [4] – a compact colour histogram representation – is used in our system.

b_1	1% - 10%	b_6	51% - 60%
b_2	11% - 20%	b_7	61% - 70%
b_3	21% - 30%	b_8	71% - 80%
b_4	31% - 40%	b_9	81% - 90%
b_5	41% - 50%	b_{10}	91% - 100%

Table 3.4: 10 bins with the same percentage capacity used in binary signature

Binary Signature which is proposed by Chitkara converts a colour histogram to a more compact format stored as the images' metadata. The process of the conversion is described below. First of all, the approach divides 1 into t bins. Each of these bins has the same capacity. For example, if t is set to 10, each of the bins denotes 10% in the format, as shown in Table 3.4.

Now, assume an image is quantized to n colours. The normalized colour histogram of this image is $H = (H_1, H_2, ..., H_n)$, where H_i is a real number denoting the percentage of pixels belonging to ith colour. The conversion from colour histogram to binary signature is then performed by using t bits to denote a colour. For colour i, its jth bit is set to 1 if H_i falls to the percentage jth bin denotes while other bits are set to 0. The signature for this image after conversion is shown as the following bit-string: $b_1^1 b_2^1 ... b_t^1 b_1^2 b_2^2 ... b_t^2 ... b_1^n b_2^n ... b_t^n$. b_j^i here denotes the jth bit of colour i. For example, there are three different images quantized to 3 colours: black, white, and gray, as shown in Figure 3.13. The binary signatures for them, according to Table 3.4, can be seen in Table 3.5.

As we can see, the binary signature of an image simply consists of several bits. On the contrary, the colour histogram of an image with n quantization colours is represented by n real numbers. Assume k bytes is needed to store one real number. Then, the storage for the histogram of an image is $n * k$ bytes. For the same image, the storage for its binary signature is just $n * t$ bits, which is obviously storage-efficient and therefore saves computing cost. For instance, using $n = 64, k = 2, t = 10$, storages for the histogram and the binary signature of an image are:

$$S_{(histogram)} = 64 * 2 = 128 \quad bytes$$
$$S_{(binarysignature)} = 64 * 10/8 = 80 \quad bytes$$

Further improvement of binary signatures can be made to save additional storage space. As we can see from Table 3.5, for each of n colours, only one bit is set to 1, while all the other bits are set to 0. So, actually the position of the bit which is set to 1 can be recorded, instead of all t bits. For example, the binary signature of Image A in Figure 3.13 can be represented in the following

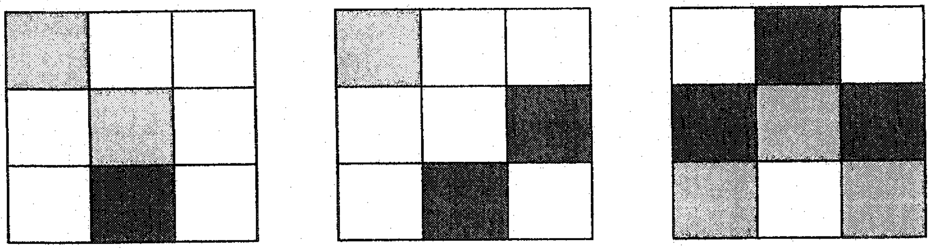


Figure 3.13: Images with three quantization colours: Black, White, and Gray

Colour	Histogram	Binary Signature									
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
Image A											
Black	11%	0	1	0	0	0	0	0	0	0	0
White	67%	0	0	0	0	0	0	1	0	0	0
Grey	22%	0	0	1	0	0	0	0	0	0	0
Binary Signature		010000000000000010000010000000									
Image B											
Black	22%	0	0	1	0	0	0	0	0	0	0
White	67%	0	0	0	0	0	0	1	0	0	0
Grey	11%	0	1	0	0	0	0	0	0	0	0
Binary Signature		001000000000000010000100000000									
Image C											
Black	33%	0	0	0	1	0	0	0	0	0	0
White	33%	0	0	0	1	0	0	0	0	0	0
Grey	33%	0	0	0	1	0	0	0	0	0	0
Binary Signature		000100000000010000000001000000									

Table 3.5: Binary Signatures for Image A, B, and C using CBA

simpler format: Image A = {2, 7, 3}. In this way, only $\lceil log_2 t \rceil$ bits are needed to store the information for each colour, instead of t bits. Using the same setup as we mentioned above, $n = 64, k = 2, t = 10$, the storage for the improved binary signature of an image, which saves 75% storage space in comparison to its colour histogram is shown as:

$$S_{(improved-binary-signature)} = 64 * \lceil log_2 10 \rceil = 32 \quad bytes$$

The binary signatures that we discussed above are converted from colour histograms by dividing 1 into t bins having the same capacity. This scheme is called Constant-Bin Allocation (CBA). There is also another scheme – Variable-Bin Allocation (VBA) – proposed in [4]. As an alternative approach to CBA, VBA is based on varying the capacity of each bin. The design of VBA is based on the fact that distances due to less dominant colours are important for efficient image retrieval. The VBA signature of an image puts more emphasis on less dominant colours than on larger dominant colours. As proved by experimental results in [4], use of the VBA signature outperforms using CBA.

Table 3.6 shows the best variable bin allocation proved by [4] when t is set to 10. This bin allocation scheme is what we used in our work. Using the VBA approach, as shown in Table 3.6, the binary signatures of Images A, B, and C in Figure 3.13 are shown in Table 3.7.

In our work, normalized LCHs are extracted from all DB images and then transferred to binary signatures, using the VBA approach, as shown in Table 3.6. All the binary signatures for the same block of DB images are stored in the same file. For example, when extracting an image's LCH by dividing it into 4 blocks, there are 4 files created. Each file contains all the signatures for all DB images of the same block. In image mosaic generating stage which will be discussed

b_1	1% - 3%	b_6	21% - 30%
b_2	4% - 6%	b_7	31% - 40%
b_3	7% - 10%	b_8	41% - 50%
b_4	11% - 15%	b_9	51% - 60%
b_5	16% - 20%	b_{10}	61% - 100%

Table 3.6: Variable bin allocation used in binary signature

Colour	Histogram	Binary Signature									
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
Image A											
Black	11%	0	0	0	1	0	0	0	0	0	0
White	67%	0	0	0	0	0	0	0	0	0	1
Grey	22%	0	0	0	0	0	1	0	0	0	0
Binary Signature		000100000000000000010000010000									
Image B											
Black	22%	0	0	0	0	0	1	0	0	0	0
White	67%	0	0	0	0	0	0	0	0	0	1
Grey	11%	0	0	0	1	0	0	0	0	0	0
Binary Signature		000001000000000000010001000000									
Image C											
Black	33%	0	0	0	0	0	0	1	0	0	0
White	33%	0	0	0	0	0	0	1	0	0	0
Grey	33%	0	0	0	0	0	0	1	0	0	0
Binary Signature		000000100000000010000000001000									

Table 3.7: Binary Signatures for Image A, B, and C using VBA

later, searching the best matching image for a tile of an original image is based on these binary signatures.

3.2.3 The image mosaic generating stage

As shown in Figure 3.6, an original image is loaded into our system and then divided into many tiles of the same size. After the original image has been tiled, each tile is regarded as an image and is processed in the same way as all DB images, as we described in the last section. First of all, a tile is quantized to the same set of colours as all DB images, then the tile is divided into blocks to extract its LCHs. After that, all LCHs are translated to binary signatures using the same bin allocation scheme as all DB images. Until now, the tile has the same number of binary signatures in the same format as all DB images. The approach we proposed to find the best matching image for a tile can then be used, based on these signatures. The basic idea of our approach is to select a set of DB images as candidate images according to the distance between their binary signatures and the input tile; then, the best matching image for the tile is selected among these candidate images. The same process goes through each and every tile of the original image. Finally, the target image mosaic is generated by substituting each tile by its best matching image. Our proposed approach to find the best matching image for a tile, which plays the most important role in the whole image mosaic generating stage, is described in detail in the next subsection.

Finding the best matching image for a tile

In order to find the best matching image for a tile, the distance between each DB image and the tile is calculated. The DB image with the smallest distance is considered as the best matching image. As we mentioned before, in the typical LCH-based image retrieval, the distance between a tile and a DB image is the sum of all absolute values of distances between two blocks at the same location, as shown below:

$$d(tile, DBimage) = \sum_{i=1}^{n} |d_i| \tag{3.4}$$

where n is the number of blocks and d_i is the distance between the ith block of a DB image and the tile. Under this scheme, a DB image can still be considered very similar to a tile even of one of its blocks is totally different to that of the tile, as long as the other blocks are very similar. This is not tolerated in our work since we need each block of the best matching image to be similar to the corresponding block of the tile, as we discussed based on Figure 3.9. In order to overcome this limitation of using the traditional LCH-based scheme, we propose a new approach which consists of two steps to find a tile's best matching image, specifically catering to our work's characteristic.

In the first step, for a specific block of an input tile, all DB images are scanned and a set of images similar to the tile with respect to that specific

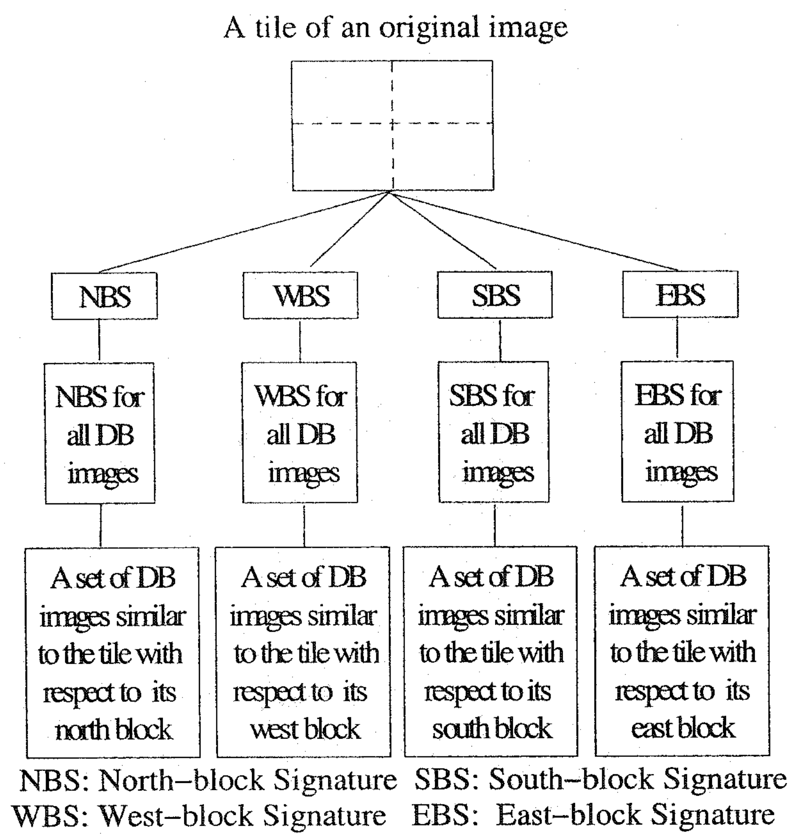

A tile of an original image
NBS
WBS
SBS
EBS
NBS for all DB images
WBS for all DB images
SBS for all DB images
EBS for all DB images
A set of DB images similar to the tile with respect to its north block
A set of DB images similar to the tile with respect to its west block
A set of DB images similar to the tile with respect to its south block
A set of DB images similar to the tile with respect to its east block
NBS: North–block Signature
SBS: South–block Signature
WBS: West–block Signature
EBS: East–block Signature

Figure 3.14: Finding the candidate images for an input tile

block is retrieved as a candidate list. For example, as shown in Figure 3.14, assume the tile and all DB images are divided into 4 blocks to extract their LCHs. The north block's signature of the tile is used to match the north block's signatures of all DB images to find the most similar images to the tile, with respect to the north block. The distance metric used when comparing two binary signatures, as proposed in [4], is simple: calculate the L1 distance between two signatures. Using this simple distance metric, a set of DB images having the smallest signature distance with respect to a specific block to the tile are selected and sorted according to their distance. These images are referred to as candidate images. The same process is performed with the remaining three blocks of the tile and selects another three sets of candidate images. The final best matching image for the input tile is selected only from these candidate images. The size of the candidate image list for each block is a parameter used in our system. It controls the number of candidate images and thus reflects the degree to which we allow a block of a DB image to be dissimilar to its corresponding block of the input tile. The effect on processing time and the quality of the target image mosaic of this parameter is discussed in detail in Chapter 4.

After obtaining all candidate images, we propose two schemes to select the best matching image among them in the second step.

Most Often Selecting Scheme (MOSS)

Obviously, if the same image appears in first place in all candidate image lists, that is a perfect matching image for the input tile. This gives us the idea that we can select the image appearing most often in all lists. We define the MOSS similarity between a candidate image and a tile as shown in Equation 3.5:

$$s_{MOSS}(tile, candidate) = \sum_{i=1}^{n} O_i \tag{3.5}$$

where n is the number of the candidate image list, $O_i = 1$ if the candidate image appears in the ith list, and $O_i = 0$ if the candidate image does not appear. We then select the image with the greatest MOSS similarity as the best matching image. For example, candidate image lists for each of 4 blocks of an input tile are shown in Table 3.8. The MOSS similarity between each candidate image and the input tile is shown in Table 3.9. As we can see, image A has the greatest MOSS similarity since it appears 3 times in 4 lists, which is the most among all candidate images. Thus, it is selected as the best matching image for the tile. The idea behind this approach is that the more times a DB image appears in candidate lists, the more blocks of this image are similar to the tile, thus the more this image matches the tile.

North_list	West_list	South_list	East_list
Image Name (distance)			
C (2)	C (3)	D (3)	B (6)
B (5)	E (6)	A (7)	F (7)
A (6)	F (9)	E (8)	A (9)

Table 3.8: 4 candidates lists.

Candidate	MOSS_similarity	Candidate	MOSS_similarity
A	1+0+1+1=3	D	0+0+1+0=1
B	1+0+0+1=2	E	0+1+1+0=2
C	1+1+0+0=2	F	0+1+0+1=2

Table 3.9: MOSS similarity between candidate images and an input tile

Candidate	DOSS_Similarity	Candidate	DOSS_Similarity
A	$\frac{1}{6}+\frac{1}{7}+\frac{1}{9}=0.42$	D	$\frac{1}{6}=0.17$
B	$\frac{1}{5}+\frac{1}{6}=0.37$	E	$\frac{1}{6}+\frac{1}{8}=0.29$
C	$\frac{1}{2}+\frac{1}{3}=0.83$	F	$\frac{1}{9}+\frac{1}{7}=0.25$

Table 3.10: DOSS similarity between candidate images and an input tile

Distance and Occurrence Selecting Scheme (DOSS)

In addition to selecting the image appearing the greatest number of times in all candidate lists as the best matching image, we try another approach which also considers the actual distance between blocks of a DB image and a tile. As we can see in Table 3.8, all candidate images in each list are sorted according to their distance to the same block of the tile. We define the DOSS similarity between a candidate image and a tile as shown in Equation 3.6:

$$s_{DOSS}(tile, candidate) = \sum_{i=1}^{n} \frac{1}{d_i} \tag{3.6}$$

where n is the number of times the candidate image appears in all candidate lists, and d_i is the distance between the candidate image and the tile in list i. We then select the image with the greatest DOSS similarity as the best matching image. The idea behind this approach is that the more often an image appears in the candidate list, the more similar it is to the input tile; at the same time, the smaller distance it has to the input tile with respect to each block, the more similar it is to the tile. However, different from MOSS, DOSS emphasis more on the small distance that a candidate image may have. That is, if two candidate images appears the same number of times in candidate lists and have the same total distance, the image with a very small distance in any candidate list is considered as the best matching image. In addition, since the DOSS is used based on candidate lists, the absence of an image in any candidate has to be considered. That is why instead of using the total distance, the reciprocal of the distance is used. We consider the similarity between an image which is not present in a candidate list and a tile to be 0. The DOSS similarities between all candidate images and an input tile are shown in Table 3.10. As we can see, image C is considered as the best matching image under this selecting scheme. Calculating the similarity between a candidate image and an input tile in this way, we consider not only the number of times a candidate image appears in all lists, but also its actual distance to the tile. How to decide the best matching image among all candidate DB images is also a parameter used in our system. A discussion of this parameter, with its experimental results is presented in Chapter 4.

3.3 Implementation Issue - Using Cache

In our work, an original image is divided into thousands of tiles, and each of these tiles is processed by the system to find its best matching image, which is a time-consuming process. Based on the analysis of our work, when an image is divided into many tiles, some of these tiles may have the exact same colour content, especially for images having a large area with the same background colours. Given the idea that tiles having the exact same colour content have the same best matching image, we try to avoid processing such tiles repeatedly, in order to speed up the process of generating an image mosaic. A caching technique is used in our system for this purpose. When the system begins processing an input tile, it goes to a cache to find whether there is any candidate tile already in the cache which has the exact same colour content as the input tile, based on their binary signatures. If the system finds one, then the input tile is considered as having the same best matching image with that candidate tile. If not, the system processes the input tile, finds its best matching image, and stores binary signatures of the input tile, as well as its best matching image, into the cache. This process goes through each and every tile of an original image during the process of generating an image mosaic. By making use of a cache, the system does not need to repeatedly scan the whole image database to find the best matching image for tiles having the same colour content, which saves a considerable amount of processing time. Some experimental results of using cache are given in Chapter 4.

Chapter 4

Experimental Results

In this chapter, different image mosaics generated by our system are evaluated according to different parameters used in the generating process. Processing time and the quality of target image mosaics are the two main characteristics we are concerned with.

4.1 Experimental Setup

As mentioned earlier, our Mosaicture methodology is designed and implemented as an image mosaic generating system. C, C-Shell script, and ImageMagic API[1] are used to implement the system under Linux OS. All the image mosaics in our experiments are generated using a PC with a 550MHz Celeron CPU and 128 Mbytes main memory.

Instead of using just one or two test images, we use a test image set consisting of 33 different images to perform the experiments, and their average result values are used for analysis. These 33 colour images are selected from the "Corel GALLERY 1,000,000" CDROM. They have different semantic content and colour distributions. For instance, some of them have a large background area with a single colour, while others have complex colour distributions. In addition, they are all of the same dimension, which ensures that the processing time and image mosaic's quality is dependent only on the different parameters

[1]http://www.imagemagick.org

Parameter	Possible Values	Default Value
(1)	33856, 13846	33856
(2)	8, 27, 64	64
(3)	4(N, W, S, E), 5(N, W, S, E, C)	4(N, W, S, E)
(4)	3, 5, 10, 30, 50	30
(5)	DOSS similarity, MOSS similarity	MOSS similarity

Table 4.1: Parameters and their default values used in our system

being used. In our experiments, 33 images are used as original images, based on which image mosaic reproductions are generated. As well, the image database used in our system is pre-processed to guarantee that all DB images also have the same dimension.

As an image mosaic generating system, the processing time for generating an image mosaic, as well as its quality, are two main concerns. Typically, a better system generates an image mosaic with higher quality using less processing time. In our system, different parameters are used to control an image mosaic generating process, which consequently affects the processing time and the quality. There are 5 parameters used in our system:

1. The size of the image database
2. The number of quantization colours
3. The scheme used to divide an image to get its LCHs
4. The number of candidate images in each candidate list
5. The approach used to select the best matching image from candidate lists

Different values or approaches of these 5 parameters are selected and used in an image mosaic generating process. When we evaluate the effect of a specific parameter, we keep other parameters at a default value. The possible choices and default values for each parameter can be seen in Table 4.1. The experimental result and analysis for each value of each parameter are discussed in detail in section 4.3.

4.2 Distance Measure

It is simple to evaluate the processing time of an image mosaic generating process using different parameter values. On the contrary, the quality of an image mosaic is difficult to evaluate in a quantitative way, since no appropriate algorithm currently exists. However, in our work, we experiment with some ideas for evaluating an image mosaic's quality in a quantitative way. Since an image mosaic is generated as a reproduction of an original image, the more similar an image mosaic is to its original image, the better the image mosaic is. So, in our work, we use the distance between an image mosaic and its original image as a criterion to evaluate the mosaic's quality. The smaller distance an image mosaic has from its original image, the better the image mosaic is. There are many approaches for calculating the distance between two images, as discussed in Chapter 3. We select two of them, as well as propose a new approach to calculate the distance between an image mosaic and its original image in order to evaluate the image mosaic's quality.

Average Pixel-to-Pixel Distance (APP Distance)

Comparing two images based on their pixels is simple to understand and implement. Under the APP Distance measure, the distance between two images, A and B, which are the same size, is calculated as the average colour difference at a pixel, as shown in Equation 4.1:

$$d(A,B) = \frac{\sum_{(x,y)=(0,0)}^{(x,y)=(m,n)} \sqrt{(r_A^{(x,y)} - r_B^{(x,y)})^2 + (g_A^{(x,y)} - g_B^{(x,y)})^2 + (b_A^{(x,y)} - b_B^{(x,y)})^2}}{m * n} \tag{4.1}$$

where (x, y) is the coordinate of a pixel, $m * n$ is the total pixel number for both images, and $r_A^{(x,y)}$ is the value of primary colour R of the pixel with the coordinate (x, y) of image A. According to this distance measure, each pair of pixels at the same location of two images is compared according to the colour value of each of three primary colours. The average distance of all pairs of pixels is deemed as the distance between two real images. For example, an original image and its two image mosaics are shown in Table 4.2, presented in the format of pixel

Image	Pixel 1(0, 0)	Pixel 2(0, 1)	Pixel 3(1, 0)	Pixel 4(1, 1)
Original Image	(23, 45, 213)	(5, 212, 37)	(145, 48, 57)	(219, 3, 4)
Image A	(34, 67, 187)	(15, 190, 42)	(196, 34, 66)	(198, 6, 12)
Image B	(123, 245, 13)	(38, 124, 231)	(45, 32, 189)	(23, 46, 165)

Table 4.2: Pixel values of an original image and its two image mosaics

colour values. All three images have 4 pixels, and the APP Distances between the original image and the other two image mosaics, according to Equation 4.1, are shown in Equation 4.2. As we can see, image A has a smaller APP Distance to the original image and thus is considered as a better image mosaic. This distance measure is based on the spatial distribution of images when calculating their distance. The colour distribution of images is not taken into consideration.

$$\begin{aligned} d(Original, MosaicA) &= 34.2 \\ d(Original, MosaicB) &= 238.77 \end{aligned} \tag{4.2}$$

GCH Distance

Different from the APP Distance measure, the GCH Distance scheme takes the colour distribution of images into consideration. As discussed in Chapter 2, GCH represents the global colour distribution of an image by counting and normalizing how many pixels belong to each colour. In our work, we choose to use the L2 distance between two GCHs to represent the distance between two images, as shown in Equation 4.3.

$$d(A, B) = \sqrt{\sum_{j=1}^{n} \left(H_j^A - H_j^B\right)^2} \tag{4.3}$$

where n is the number of quantization colours, and H_j^A is the colour histogram for jth colour in image A. For example, GCHs for an original image O and its two image mosaics under 4 quantization colours, A and B, are shown below:

$$\begin{aligned} H^O &= \{15\%, 4\%, 45\%, 36\%\} \\ H^A &= \{10\%, 5\%, 45\%, 40\%\} \\ H^B &= \{45\%, 10\%, 5\%, 40\%\} \end{aligned}$$

According to the L2 distance measure that we discussed in Chapter 2, the distance between the original image and its image mosaics are:

$$\begin{aligned} d(O, A) &= 0.065 \\ d(O, B) &= 0.51 \end{aligned} \tag{4.4}$$

As we can see, A is considered as a better image mosaic since it has a smaller GCH distance to the original image.

Multi-level Colour Histogram Distance (MLCH Distance)

An image mosaic has an intended visual effect. The content of the whole image comes out clearly only when it is looked at from many steps away. Due to this characteristic of image mosaics, we propose a new approach for evaluating an image mosaic's quality. This approach is based on LCH distance, while considering the special characteristics of image mosaics. The basic idea is the following: when two image mosaics are looked at from far away, their difference probably cannot be seen clearly, but when they are looked at from a closer distance, their difference is more clear. I.e., the closer we are to two image mosaics, the more difference we can see between them. What we do in our MLCH Distance measure is the following: we first calculate the GCH distance between the images according to Equation 4.3, we then tile both images and obtain their average LCH distance. Finally, we use the difference between these two distances as the distance between the original image and its image mosaic. The smaller this distance is, the better an image mosaic is. The idea behind this approach is that the better an image mosaic is, the smaller difference there is between looking at the image from different distances. In our work, we tile an image mosaic, as well as its original image, into 5x5 blocks to get the average

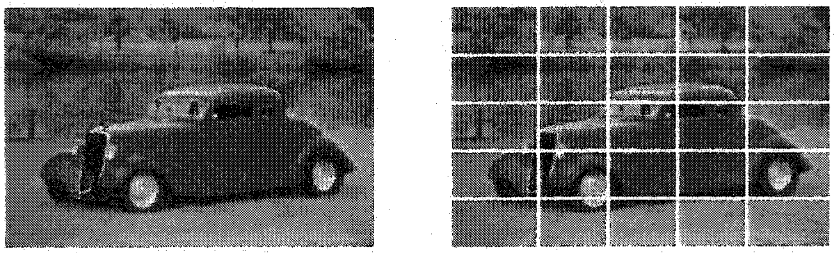


Figure 4.1: The left-hand image is tiled into 5x5 blocks to extract LCHs

LCH distance, as shown in Figure 4.1. Finally, we use the difference between the GCH distance and the average LCH distance under 5x5 blocks as the final distance. Equation 4.5 shows how we calculate the distance between an original image and its image mosaic A.

$$d_{MLCH}(O,A) = d_{LCH}(O,A) - d_{GCH}(O,A) \tag{4.5}$$

where $d_{LCH}(O,A)$ is shown below:

$$d_{LCH}(O,A) = \frac{\sum_{i=1}^{k*l} \sqrt{\sum_{j=1}^{n} (H_j^A(i) - H_j^B(i))^2}}{k*l} \tag{4.6}$$

where k and l are the block numbers of an image being divided into on length and on width, respectively, n is the number of quantization colours; and $H_j^A(i)$ is the colour histogram for jth colour in ith block of image A.

Colour Histogram	Original Image	Image Mosaic A
GCH	{20%, 30%, 15%, 35%}	{10%, 25%, 35%, 30%}
Block 1's LCH	{30%, 10%, 20%, 40%}	{10%, 20%, 20%, 50%}
Block 2's LCH	{20%, 10%, 20%, 50%}	{5%, 30%, 20%, 45%}
Block 3's LCH	{15%, 40%, 10%, 35%}	{15%, 45%, 40%, 0%}
Block 4's LCH	{15%, 60%, 10%, 15%}	{10%, 5%, 60%, 25%}

Table 4.3: GCHs and LCHs for an original image and its image mosaic

For example, the GCHs as well as the LCHs under 2x2 blocks of an original image, and its image mosaic under 4 quantization colours, are shown in Table 4.3. According to Equation 4.3, the GCH distance between the original image and image mosaic A is:

$$d_{GCH}(O, A) = 0.23$$

As well, according to Equation 4.6, the average LCH distance between them is:

$$d_{LCH}(O, A) = 0.425$$

Then according to Equation 4.5, the MLCH Distance between the original image and image mosaic A is:

$$d_{MLCH}(O, A) = 0.425 - 0.23 = 0.195$$

Three distance measures have been discussed up till now in this section. We choose APP and GCH distance measures, since they consider the spatial distribution and colour distribution of images, respectively. The new MLCH distance measure we proposed is based on human perception of images. According to these distance measures, the smaller the distance an image mosaic has to its original image, the better the image mosaic is. Image mosaics' quality will be analyzed based on these three distance measures. In addition, an analysis of our new distance measure is given in section 4.4.2.

4.3 Experimental results and analysis

In this section, all experimental results are given and analyzed. When generating an image mosaic, we divide the original image into $64*64$ tiles, which guarantees that the dimension of each tile is the same as the DB images, and, the quality of image mosaics generated is acceptable. There are 5 parameters used in our system to control an image mosaic generating process, as shown in Table 4.1. Processing time and image mosaics' quality are analyzed according to each of these 5 parameters. The processing time for generating an image mosaic includes quantizing the original image, dividing it into many tiles, extracting LCHs from each tile, translating LCHs to binary signatures, matching these signatures to all DB images, and selecting the best matching one. As to the quality of an image mosaic, since this is the first time that the quantitative algorithms are used, we use all three measures we discussed in last section to evaluate the quality of image mosaics together. In addition, in order to make all results using each measure comparable, we normalize all distances to be in the range of 0% - 100%. 0% means the image mosaic is exactly the same as the original image, while 100% means the image mosaic and its original image are totally different.

4.3.1 The size of the image database

Processing Time

The size of the image database affects the processing time for generating an image mosaic directly. Obviously, the larger an image database is, the more processing time the system spends to generate an image mosaic – since more DB images are searched in the image mosaic generating process. In our experiments, we compare the average processing time for generating an image mosaic based on two image databases, which have different numbers of images. One image

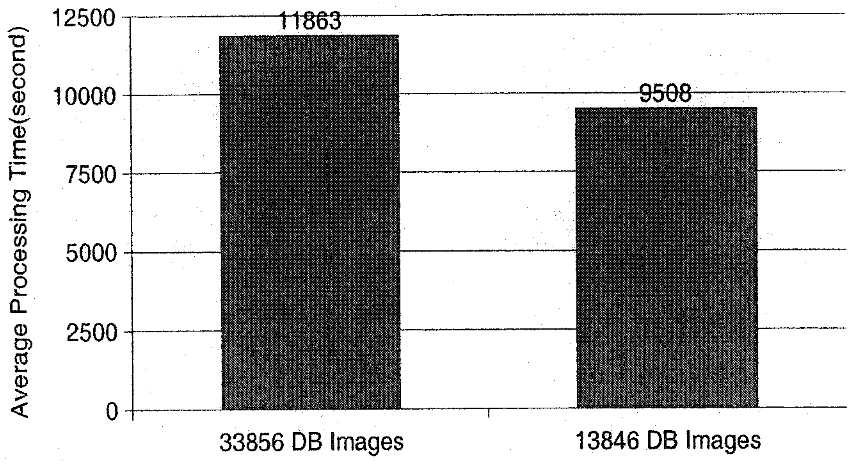


Figure 4.2: Processing time – Different size of image database

database has 33856 DB images while the other one has 13846 DB images. The size and dimension of images in both image databases are the same. When comparing the processing time, we keep other parameters at the default value, as shown in Table 4.1. As we can see from the result chart, Figure 4.2, the system uses 11863 seconds, on average to generate an image mosaic based on the larger database, and, 9508 seconds based on the smaller database. In other words, the system spends 20% less time to generate an image mosaic based on the smaller database. Further analysis shows that instead of a linear relationship, this 20% processing time is saved using an smaller image database which has 60% fewer images than the larger one. This result can be used to encourage using larger image databases when processing time is not such a critical factor considered in a system.

Image Mosaic's Quality

When an image mosaic is being generated, its original image is first divided into many tiles. For each tile, the system searches the image database and selects the best matching image for it, based on its binary signatures. Basically, the more DB images we have, the better the DB images similar to the tile that can

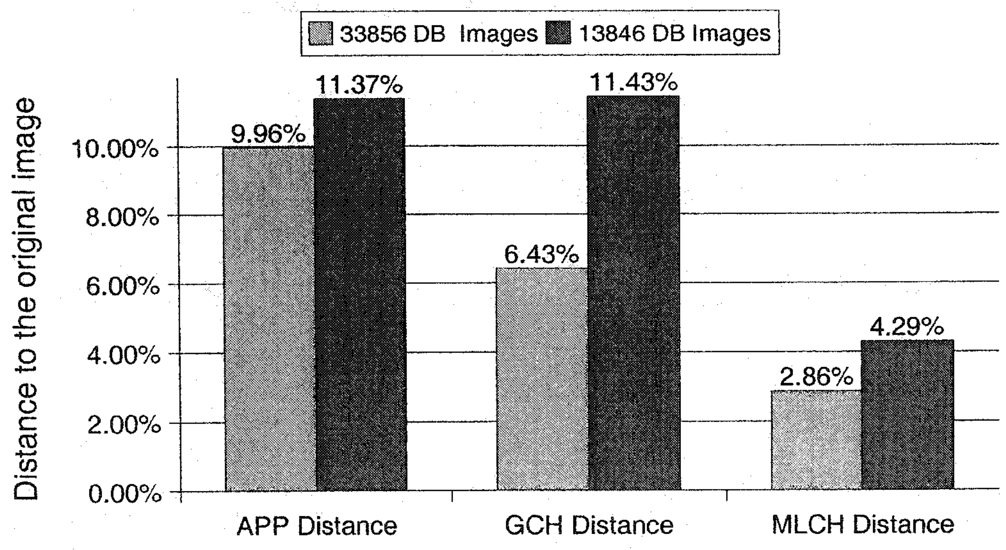


Figure 4.3: Quality comparison – Different size of image database

be selected into the candidate list, thus the better "best matching" image we can find for the tile. Figure 4.3 shows the experimental results. The average distance between 33 image mosaics and their original images under 3 distance measures – APP Distance, GCH Distance, and MLCH Distance – are put together in one figure. As we can see, under all distance measures, image mosaics generated based on the larger database have smaller distances to their original images. In particular, for images with a large area of a single colour, many divided tiles have the same colour content. If a DB image having the most similar colour content to these tiles only appears in the larger image database, the quality of the target image mosaic based on this larger image database will obviously be better.

	APP Distance	GCH Distance	MLCH Distance
Difference	1.41%	5%	1.43%

Table 4.4: Difference between image mosaic's quality using 3 different distance measures

However, the difference between the image mosaics' quality based on the different size of the image databases is small, as shown in Table 4.4. Since an image mosaic's quality is evaluated according to its similarity to the original image, the absolute difference between two image mosaics' normalized distance

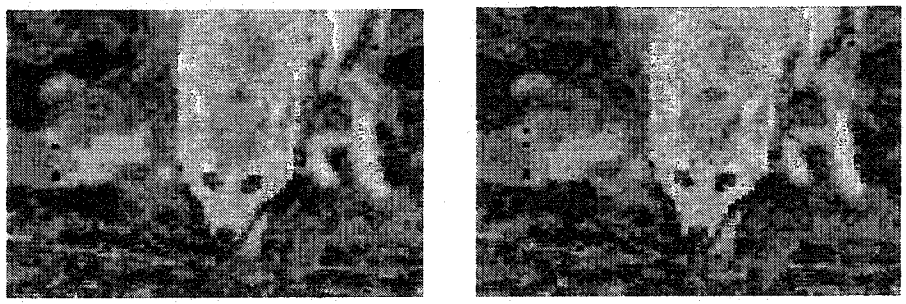


Figure 4.4: The right-hand mosaic which is generated using the bigger imageDB is better but not so obviously

to the original image is regarded as the difference between their quality. As we can see from Table 4.4, the greatest difference between image mosaics' quality is 5% under GCH Distance, while it is only around 1.4% under other distance measures. Figure 4.4 shows two such image mosaics generated based on different number of DB images. We cannot easily tell which one is definitely better, even though we can see that there is some more noise on the left one. In addition, the image mosaics generated using the smaller database still have a similar visual content to the original images according to human perception. This result tells us that an image database having 13846 images is large enough. In comparison with the difference in processing time, which is 20% between the larger and the smaller image database, we believe the smaller image database contains enough DB images to generate a good image mosaic. Anyway, if the quality of image mosaics is the main concern of users, large image databases can be used to generate better image mosaics, due to the non-linear relationship between DB image numbers and processing time.

4.3.2 The number of quantization colours

The number of quantization colours determines how much colour information about an image is encoded in its metadata. Basically, the greater the number of colours an image is quantized to, the less colour information is lost. For example, when the quantization colour is 64, that means there are 4 different colour levels in each of three primary colours under the uniform quantization scheme. When the quantization colours are set to 8, there are only 2 different colour levels left. When more colour information is stored in the metadata, more storage is used and more time is needed to compare this information with other data. The storage for binary signatures of an image using different number of quantization colours is shown in Table 4.5.

Processing Time

Since more colour information is encoded in the metadata of DB images, as well as an original image, using a higher number of quantization colours,

Quantization Colours	Storage Requirement for an image
64	$64 * 4 * 4$ bits = 128 bytes
27	$27 * 4 * 4$ bits = 54 bytes
8	$8 * 4 * 4$ bits = 16 bytes
The image is divided into 4 blocks to get LCHs	
4 bits are used for each colour in an image's binary signature	

Table 4.5: Storage for an image's metadata using different colours

more work is needed to compare these metadata. Figure 4.5 shows the average processing time for generating image mosaics for 33 test images using different quantization colours. As we can see, the more quantization colours used, the more time the system spends to generate an image mosaic. Note that, the difference between processing times using different quantization colours is very dramatic. For instance, generating an image mosaic using 27 quantization colours saves 62% processing time in comparison with the same process using 64 colours. The result tells us that the processing time for generating an image mosaic is heavily dependent on the number of quantization colours used.

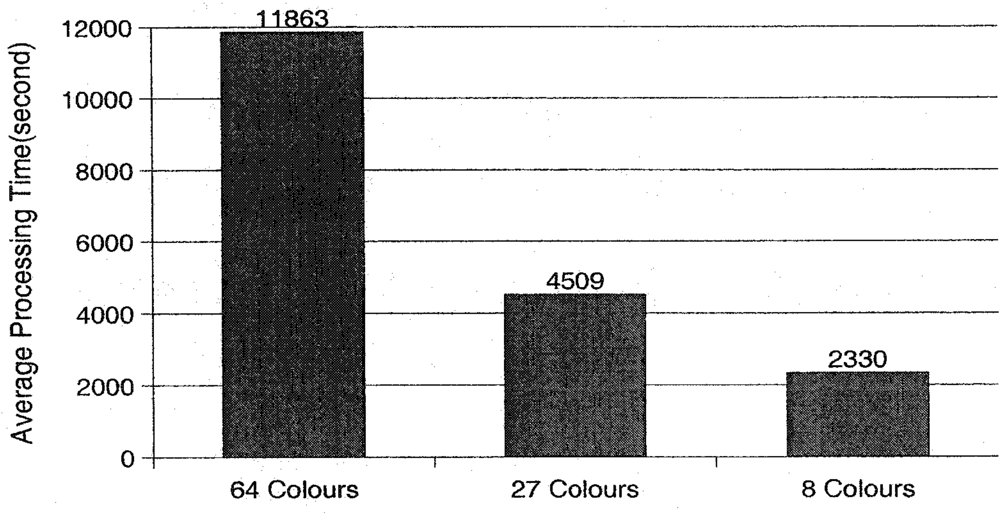


Figure 4.5: Processing time – Different quantization colours

Image Mosaic's Quality

As discussed, the larger number of quantization colours used, the more processing time is needed for generating an image mosaic. However, since more colour information of an image is encoded, its metadata can represent the real image better. For example, when an image is quantized to only 2 colours, many different colours in the original colour space are combined into a single colour. However, if 64 quantization colours are used, they can probably still be represented by different colours in the new colour space. A larger number of quantization colours helps our system to differentiate images more precisely. Thus, better "best matching" images for input tiles can be selected. Figure 4.6 shows the experimental results of comparing image mosaics' quality. As we can see from the figure, in general, the more quantization colours used, the better the image mosaics are. Further analysis shows that the quality difference between image mosaics using 27 and those using 8 quantization colours is relatively small. Meanwhile, image mosaics using 64 colours have obvious smaller distance than those using the other two numbers of quantization colours. For example, under the GCH Distance measure, the distance of image mosaic generated using

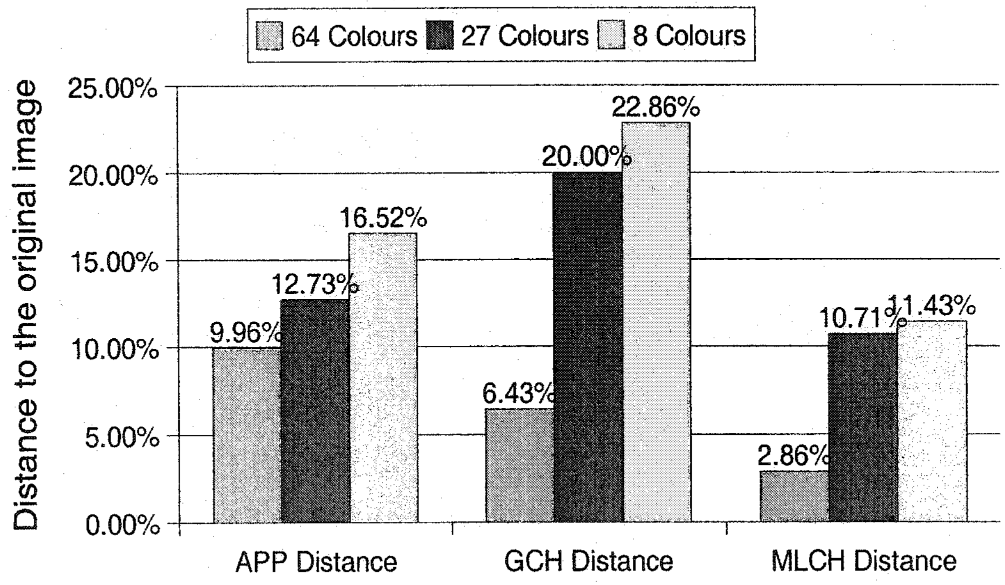


Figure 4.6: Quality comparison – Different quantization colours

64 colours is 16.43% smaller than that using 8 colours. This shows that, while using 8 and 27 quantization colours can only represent an image at a low level of accuracy, the metadata using 64 colours represents the real image the best.

4.3.3 The scheme used to extract LCHs

Processing Time

As one of the parameters used in our image mosaic generating system, two different schemes for extracting LCHs from an image are experimented with. In our system, we divide an image into different numbers of blocks to extract its LCHs. One scheme is extracting the image into 4 blocks – North, West, South, and East – while the other one adds an additional Central block. According to our methodology, for the purpose of finding the best matching image for an input tile, a candidate image list is first selected for each block. Under the 4-block scheme, 4 binary signatures of a tile of an original image are processed, and 4 candidate list are selected. The final best matching image is selected from these 4 candidate lists of images. Under the 5-block scheme, one additional binary signature is processed, and thus one more candidate list is considered when selecting the final best matching image. Therefore, the processing time

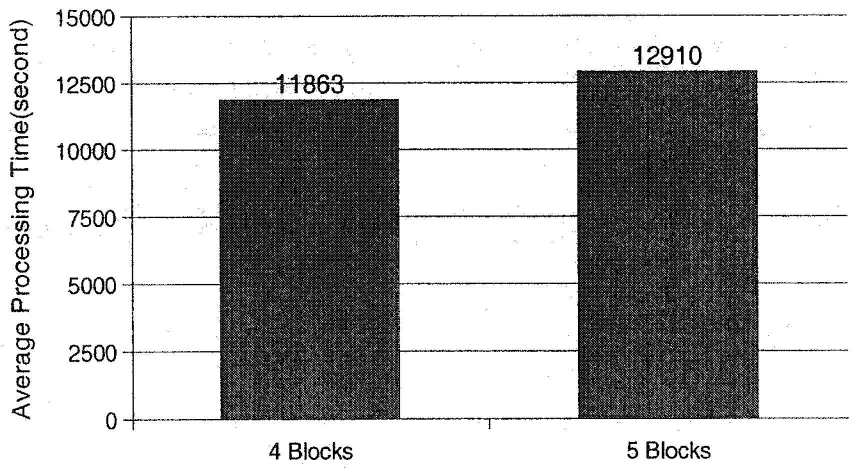


Figure 4.7: Processing time – Different schemes to extract LCHs

for generating an image mosaic under the 5-block scheme is greater than under the 4-block scheme. Figure 4.7 shows the average processing time for generating an image mosaic under different LCH extracting schemes. As we can see, the system spends 11863 seconds to generate an image mosaic under the 4-block scheme which is smaller than 12910 seconds used under the 5-block scheme. However, in comparison with the 5-block scheme, the system only saves 8.1% of processing time. The reason is that under the 4-block scheme, each block of an original image has more pixels than the same block under the 5-block scheme. Thus, more time is needed when calculating the histogram of a block. The average numbers of pixels for a block under two different schemes are shown in Equation 4.7,where n is the total number of pixels of an image.

$$\begin{aligned}\mathrm{EachBlock'sPixelNumber}_{(4-\mathrm{block})} &= \frac{1}{2} * n \\ \mathrm{EachBlock'sPixelNumber}_{(5-\mathrm{block})} &= \frac{n/9 + 4 * n/3}{5} = \frac{13}{45} * n \qquad (4.7)\end{aligned}$$

Image Mosaic's Quality

In CBIR systems, different schemes used to divide an image into blocks in order to extract LCHs affects the result of an image retrieval process. Typically, the greater the number of blocks an image is divided into, the more image information is encoded into the metadata, thus the better the result of image retrieval is. According to this, using the 5-block scheme to extract images' LCHs should also help the system to generate better image mosaics than use of the 4-block scheme. This is true, as shown in Figure 4.8. Under GCH and MLCH distance measures, using the 5-block scheme is better. However, the difference shown in the result is small. Also, under the APP Distance, the image mosaic generated under the 4-block scheme is even a little bit better than that achieved by using the 5-block scheme. The reason is that finding a good best matching image for each tile is not the only factor that determines the quality of the final image mosaic. In our system, hundreds or thousands of images are put together to build up an image mosaic. So, the relationship between adjacent images is as well important. Typically, a good image mosaic requires that adjacent images have smooth colour conversions, especially at their edges. Under the

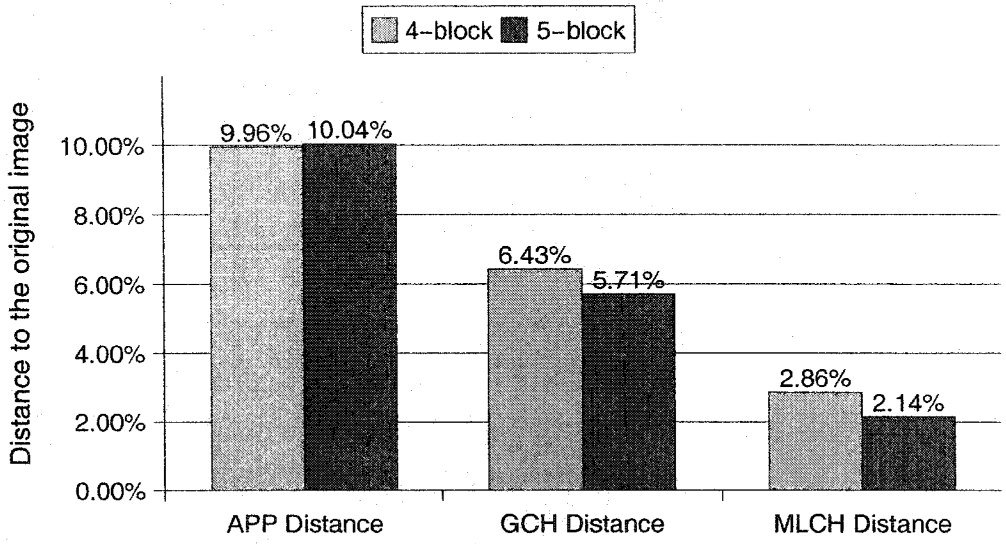


Figure 4.8: Quality comparison – Different schemes to extract LCHs

5-block scheme, the central part of an image plays a more important role when retrieving its best matching image than the 4-block scheme, which may result in not so good conversions between adjacent images. On the whole, the results show that the scheme of extracting images' LCHs actually does not affect the image mosaics' quality very much in our system. However, we select the 5-block scheme as the better choice since the image mosaics using this scheme has smaller distance under both GCH and MLCH distance measures.

4.3.4 The size of the candidate image list

Processing Time

This parameter, the size of the candidate image list, also affects the processing time for generating an image mosaic. Figure 4.9 shows the experimental results. As we can see, the more images a candidate list has, the more processing time is needed to generate an image mosaic. According to our methodology, a bigger list size means more DB images are selected as candidate images for an input tile, which in turn costs more processing time. Besides, the system also needs more time to calculate the distance of each image in the candidate lists in

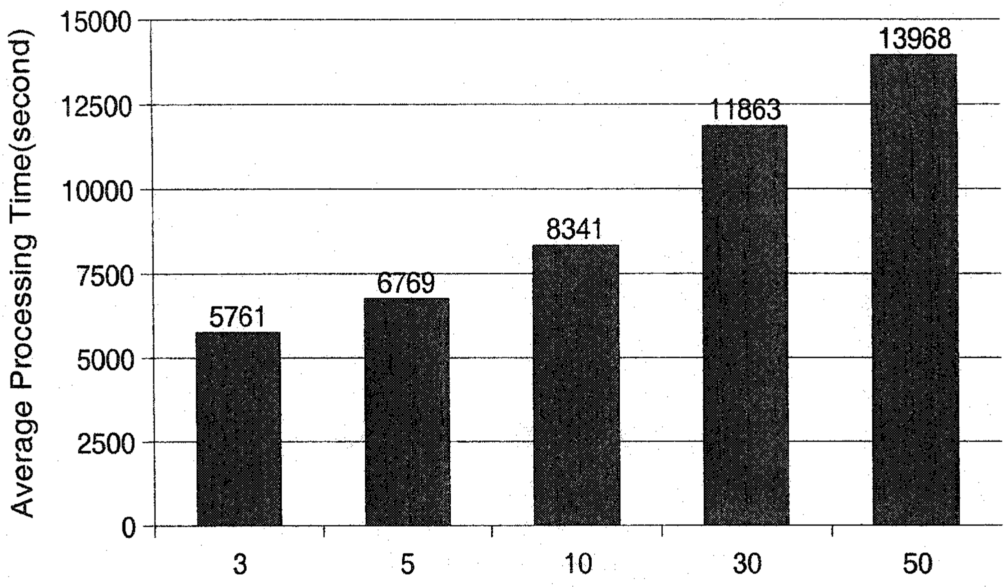


Figure 4.9: Processing time – Different size of candidate image list

order to select the best matching image. Since the processing time of these two procedures in our system are affected by the size of the candidate image list in the way we discussed, the bigger a candidate list is, the more processing time is needed to generate an image mosaic.

Image Mosaic's Quality

Different from how it affects the processing time to generate an image mosaic, the size of a candidate image list does not determine the quality of the image mosaic. Figure 4.10 shows the experimental results. As we can see, using a bigger candidate image list may generate either better or worse image mosaics. For example, under the GCH distance measure, the image mosaic generated using 5 as the list size is better than that using 50. Meanwhile, it is worse than the image mosaic generated using list size 3. More analysis of the experimental results show us that, under any distance measure, the difference among the image mosaic's quality using different list sizes is very small, as shown in Table 4.6.

The reason behind this is that the best matching image selected for a tile is not definitely better when more images are selected into each candidate list.

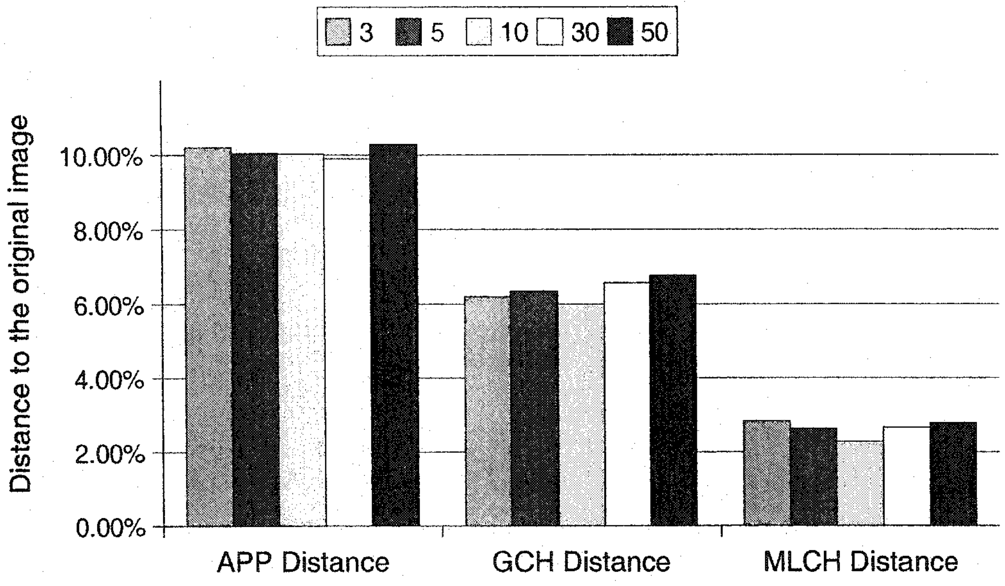


Figure 4.10: Quality comparison – Different size of candidate image list

	APP Distance	GCH Distance	MLCH Distance
Biggest Difference	0.39%	0.78%	0.55%

Table 4.6: Difference among image mosaics using different list sizes

For example, suppose we use the MOSS similarity to select the best matching image among 4 candidate image lists. Table 4.7 shows the four lists when the size is set to 3. As we can see, candidate image A appears twice, which is the most often among all candidate images, so it is selected to be the best matching image for the input tile.

	North_List	West_List	South_List	East_List
1	B	D	G	I
2	C	E	H	J
3	A	F	A	K

Table 4.7: Candidate image lists with size 3

	North_List	West_List	South_List	East_List
1	B	D	G	I
2	C	E	H	J
3	A	F	A	K
4	L	L	L	M
.	.	.	.	.
.	.	.	.	.
50	.	.	.	.

Table 4.8: A good example of setting list size to 50

	North_List	West_List	South_List	East_List
1	B	D	G	I
2	C	E	H	J
3	A	F	A	K
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
50	L	L	L	M

Table 4.9: A bad example of setting list size to 50

Now we set the list size to 50. If the candidate images are selected as shown in Table 4.8, then candidate image L is selected as the best matching image. As we can see, image L appears 3 times in all four lists and in addition, the positions of it are all very near the top of the lists. In this case, we believe image L is more similar to the input tile than image A. However, if the candidate images are selected as shown in Table 4.9, the result is opposite. This time, candidate image L appears at all final positions in candidate lists, which means its distance to the input tile is great. However, it is still selected as the best matching image due to its 3 appearances on all 4 candidate lists. In this case, image L is more likely not as good as image A as the best matching image for the input tile.

In conclusion, we believe list size 3 to be the best choice since it cost the system the least processing time without an obvious drop in quality.

4.3.5 The approach used to select the best matching image

Processing Time

As one of our system's parameters, the approach used to select the best matching image among candidate image lists does not affect the processing time at all. Figure 4.11 shows the experimental results. As we can see, the average processing time for generating an image mosaic using different approach for selecting the best matching image is almost the same. The reason is that, unlike other parameters, the only process affected by this parameter are memory operations which are performed after all the candidate images are selected. Therefore, the processing time changes little, if at all, when different approach of this parameter is used.

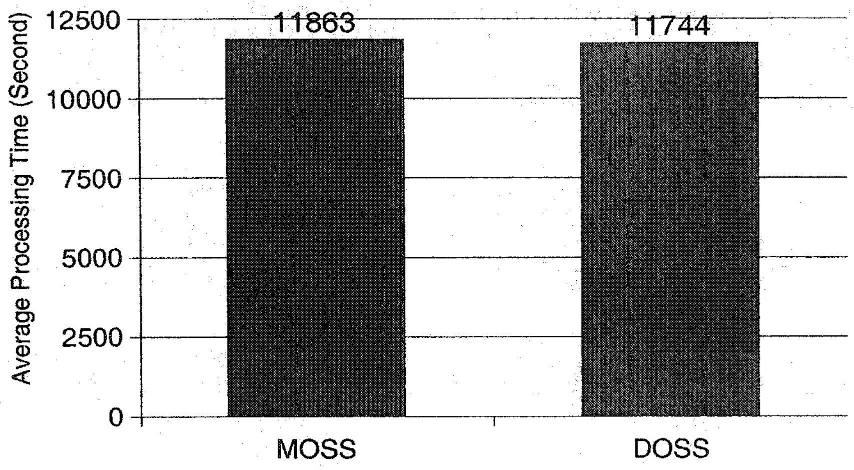


Figure 4.11: Processing time – Different best matching image selection

Image Mosaic's Quality

Even though it does not affect the processing time, the approach used to select the best matching image does indeed affect the quality of the image mosaic. As discussed in Chapter 3, there are two approaches used in our system: MOSS, which selects the image with most occurrences in all candidate lists as the best matching image; and DOSS, which takes the distance of candidate images into consideration. Figure 4.12 shows the experimental results. As we can see, the image mosaic generated using MOSS always has a smaller distance to the original image. This shows that more blocks of an image being similar to a tile is more important than the smaller total distance of it to the tile. In other words, an image with more blocks similar to a tile should be selected as the best matching image rather than the one with fewer similar blocks, but having a smaller total distance.

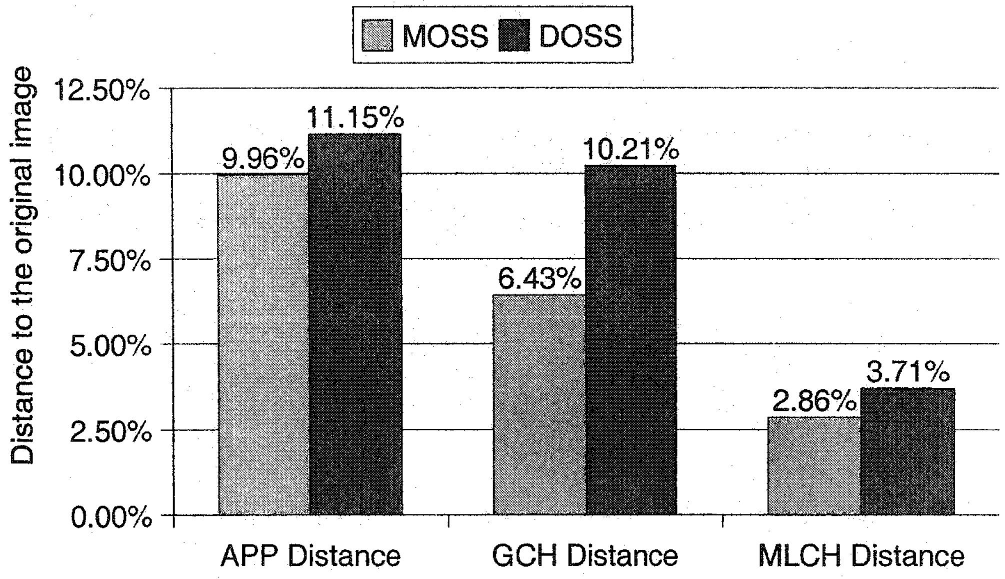


Figure 4.12: Quality comparison – Different best matching image selection

4.4 Conclusion

4.4.1 The effects of five parameters

Our image mosaic generating system has been experimented according to the parameters used. After analyzing the experimental results, we know that each of these 5 parameters affects the processing time and the quality of an image mosaic to some degree. However, different parameters play different roles in the system. For example, with respect to the processing time, the number of quantization colours affects the system the most, while the approach of selecting the best matching image among candidate images does not play an important role. The degree of effect on the processing time and image mosaics' quality of different parameters are shown in Tables 4.10 and 4.11.

Parameters	Effect on Processing Time
Quantization Colours	Greatest
Size of Candidate List	Second
Size of Image DB	Third
LCH Extraction	Small
Best Matching Image Selection	Small

Table 4.10: The effect of different parameters on processing time

Parameters	Effect on image mosaic's quality
Quantization Colours	Greatest
Size of Image DB	Second
Best Matching Image Selection	Third
LCH Extraction	Small
Size of Candidate List	Small

Table 4.11: The effect of different parameters on image mosaic's quality

Parameters	Best Value or Approach
Quantization Colours	64 Colours
Size of Image DB	13846 DB Images
Best Matching Image Selection	MOSS Similarity
LCH Extraction	Extract to 5 blocks
Size of Candidate List	3

Table 4.12: The best value or approach for each parameter

In addition, according to our analysis of the results, the best choice of values or approaches we tested for each parameter can be made considering the processing time and image mosaic's quality together. For example, even though using 64 quantization colours costs the system more processing time to generate an image mosaic, we still believe that it is the best choice among all quantization colours tested, because of the improvement in the image mosaic's quality. The best choice for each parameter selected by us is shown in Table 4.12. However, according to different focuses of different image mosaic generating systems, other values or approaches may be selected as the best choice catering the system's need.

4.4.2 Analysis of MLCH measure

Since this is the first time that quantitative measures are used to evaluate the quality of image mosaics, we analyze the experimental results based on all three distance measures together – APP distance, GCH distance, and MLCH distance. In our experiments, all the three distance measures we tested show the consistent results with each other when evaluating the quality of image mosaics In addition, the comparison results of the quality of different image mosaics are also consistent with our perception.

Among these three measures, the MLCH distance measure is the one we proposed as an original idea to evaluate the quality of image mosaics, based on human perception. As discussed earlier, the MLCH distance measure is based on two hypotheses:

- The closer an image mosaic is looked at, the more obviously the visual difference between it and its original image can be seen.
- A better image mosaic has a smaller increase on its distance to the original image when being looked at increasingly smaller distance.

The experimental results obtained validated our hypotheses. In order to explain the characteristic of MLCH distance measure more clearly, as an example, we redo the experiment to get the MLCH distance between original images and image mosaics generated using 27 and 64 quantization colours. This time, instead of just obtaining the LCH distance at 5x5 level, we show the LCH distance at 10 different dividing levels in Figure 4.13. As we can see, for using both numbers of quantization colours, image mosaics generated always have a greater distance to original images when more blocks are tiled to obtain average LCH distance. This validate our first hypothesis, which is actually very similary with the method proposed in Nicholas Tran's paper to evaluate an image mosaic's quality. In

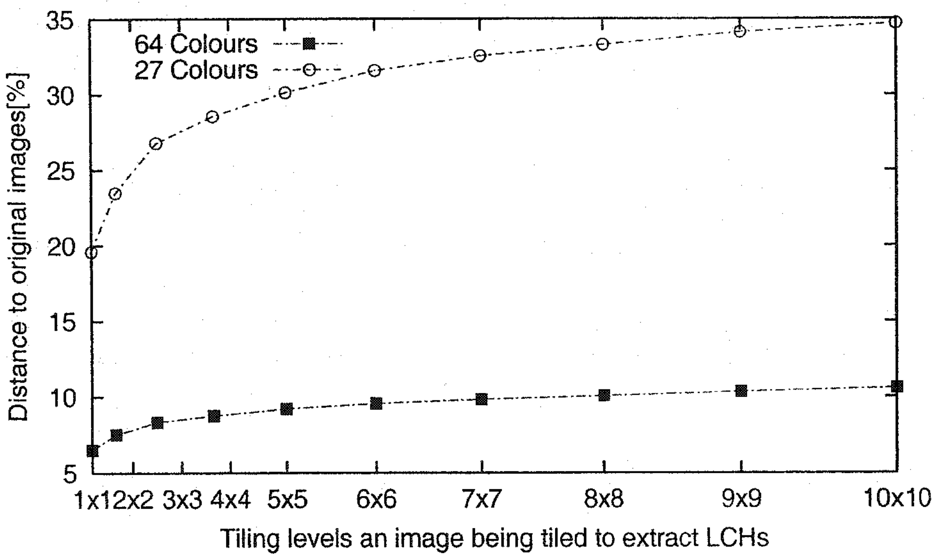


Figure 4.13: LCH distance of image mosaics using 27 and 64 quantization colours at different LCH extracting level

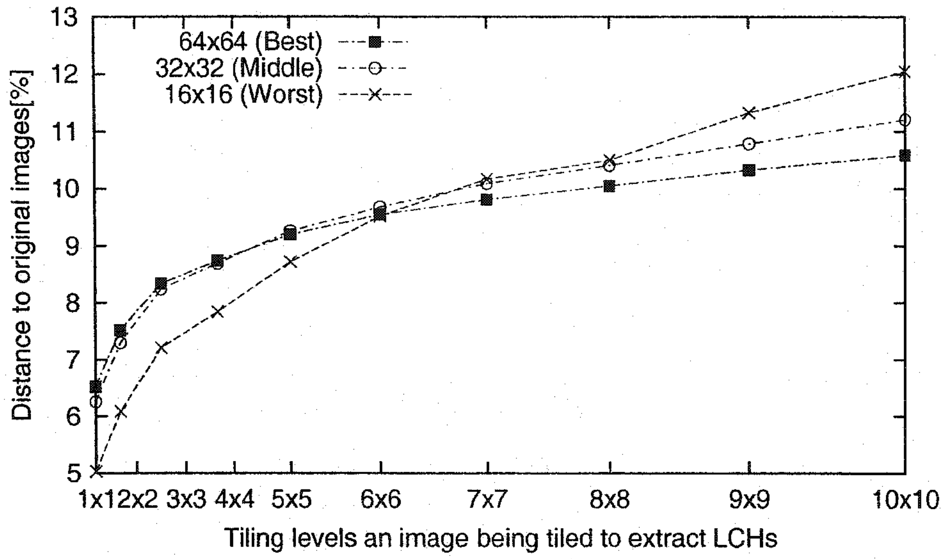


Figure 4.14: LCH distance of image mosaics using different tiling numbers at different LCH extracting levels

addition, the better image mosaics, which are generated using 64 quantization colours, have smaller increases on average LCH distance to original images with the dividing level being increased. This also validate our second hypothesis.

In addition, we also conducted an additional experiment for the sole purpose of comparing GCH and MLCH distance measures. In this experiment, we generate three sets of image mosaics when their original images are divided into different number of tiles – 64x64, 32x32, and 16x16 – which affects the quality of image mosaics obviously. Based on our perception, an image mosaic generated when the original image is divided into greater number of tiles also has a better quality. However, the GCH distance measure failed to reflect the consistent result in this experiment. As shown in Figure 4.14, the average LCH distance when the dividing level is 1 is actually the GCH distance between image mosaics and original images. As we can see, the worse image mosaic has a smaller GCH distance which conflicts with human perception. In addition, even if the LCH distance measure, which is not tested in our former experiments, is used, the comparison result of the quality of image mosaics is not consistent as discussed next. As we can see from Figure 4.14, the better image mosaic has a smaller LCH distance at a high granularity level, while having a greater LCH distance at a low granularity level. However the MLCH distance measure shows a consistent result. The better image mosaic has a smaller increase on LCH distance to the original image with the dividing level being increased. This is shown in Figure 4.14, as the better image mosaic having a LCH distance curve with a smaller slope. Therefore, in this experiment, our proposed MLCH distance measure outperforms the GCH as well as the LCH distance measure.

4.5 Cache usage analysis

According to our methodology, an original image is first divided into hundreds or thousands of tiles; then an image mosaic is generated by replacing these tiles by their best matching images. As discussed in Chapter 3.3, many of these tiles may have the exact same binary signatures. Cache technique is used in our system to reduce the processing time by avoiding processing tiles whose binary signatures are the same as some other tiles. As shown in Table 4.13, among 4096 tiles that an image is divided into, there are an average 2077 tiles having exact binary signatures with some other tiles. Cache technique is used to prevent the system from searching the whole image database to find best matching images for these tiles. The better a cache technique is, the fewer tiles having the same signatures as others will be processed by the system. In our experiments, a cache which could be used exactly 2077 times is called a perfect cache. The cache usage we mention hereafter is the percentage of cache hit in comparison with the perfect cache. Some factors must be considered in order to achieve a high cache usage, and these are discussed below.

The cache used in our system is based on a FIFO scheme. That means,

Total tiles of an original image	Tiles having same signatures as others
4096 (64x64)	2077

Table 4.13: Half of total tiles having the same binary signatures as others

Cache Size	128	64	32	16
Cache Usage	96%	94%	93%	92%

Table 4.14: Cache usage of different cache size with comparison to a perfect cache

when the cache is full, the oldest tile in the cache is removed so that a new tile can be stored. Obviously, the larger a cache is, the more images can be stored in the tile; as well the higher cache usage will be. Different cache sizes are experimented with, as shown in Table 4.14. As we can see, the larger the cache is, the higher cache usage is achieved. However, the cache usage is already very high and the differences among these cache sizes are small.

A cache cannot be perfect since tiles are being removed when the cache is full. A tile which is removed may be the only one in the cache at that time having the same signatures as the next input tile. Therefore, keeping the tiles having the similar content as nearer to each other as possible is probably another way to achieve a higher cache usage. We experimented with changing the tiling order of an original image for this purpose, due to the possibility of tiles having the same content are in some rectangular area. However, the result shows that the tiling order does not affect the cache usage a great deal. The three different tiling orders are shown in Figure 4.15. Under tiling order 2 and 3, an original image is divided into two and four sub images, respectively; and for each sub

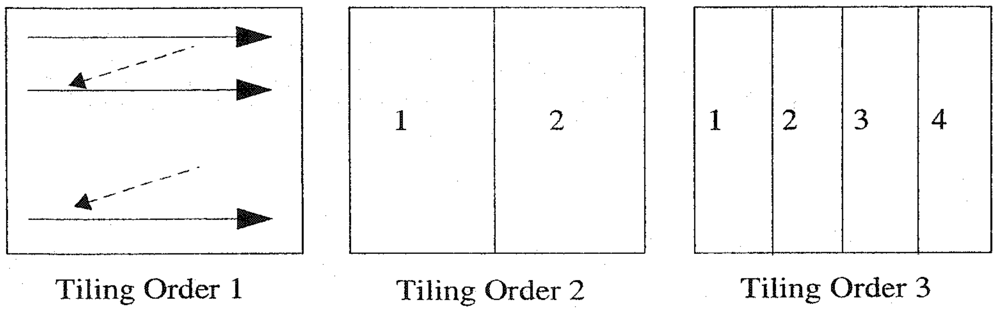


Figure 4.15: Three different tiling orders

	Tiling Order 1	Tiling Order 2	Tiling Order 3
Cache Usage	94.1%	94.9%	95.7%

Table 4.15: Cache usage using different tiling orders in comparison to a perfect cache

image, the tiling order is the same as tiling order 1. The cache usage using these three tiling orders for cache size 64 is shown in Table 4.15. Finally, the real processing time using cache size 64 with tiling order 3 is shown in the Figure 4.16 (All parameters are set to their default values when generating image mosaics using a cache). The system uses 20% less processing time on average to generate an image mosaic than no cache being used. Note that, even though the cache usage is already very high in our system (around 95%), which means almost half number of total tiles of an original image are not processed by the system to find the best matching image, the real processing time saved is only 20% instead of 50% which is supposed to be. The reason is that the process for searching the image database, and finding the best matching image for a tile is not the only work needed to be done for generating an image mosaic. Other processes are also included when the processing time is recorded, for example, dividing the

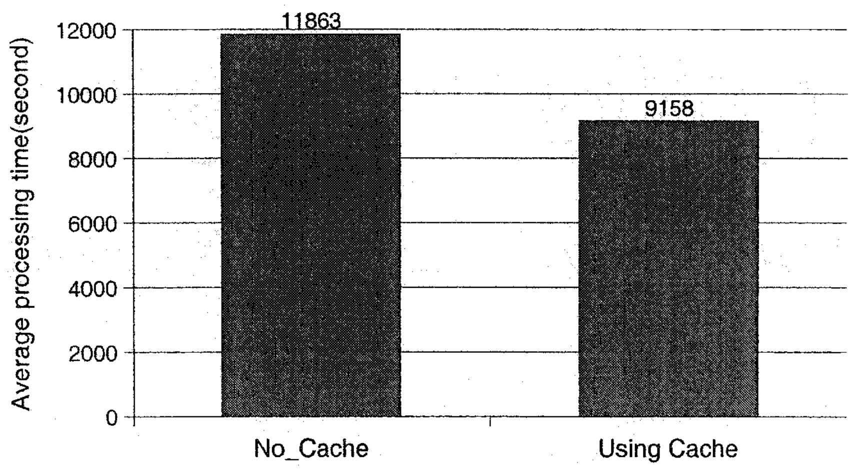


Figure 4.16: Processing time using cache in comparison to no cache being used

original image into thousands of tiles, extracting colour features of a tile and translating its colour histograms to binary signatures. That's why even a high usage cache can not save so much processing time.

There are also many other methods can be considered when using a cache. For example, in our system, an input tile is not processed only if another tile having the exact same binary signatures exists in the cache. More processing time can be saved considering tiles with small distance as the same. However, the quality of the image mosaic is then supposed to decrease, since different tiles use the same best matching image. The relationship between an image mosaic's quality and the degree of difference to which two tiles can be regarded having the same best matching image is beyond the scope of this thesis and can be put in the future work.

Chapter 5

Conclusion and Future work

5.1 Conclusion

The main contribution of this thesis is the design and implementation of an image mosaic generating system, which is called Mosaicture, and, the investigation of different parameters' roles in the system. In our system, an image mosaic is generated as a reproduction of an original image. Before an image mosaic can be generated, the image database is pre-processed in order to extract the colour features of all DB images. Colour histograms are used to represent these features, which are then translated to binary signatures. Binary signature is a kind of compact colour feature representation used in our system which saves a great deal of space when storing the images' metadata as well as speeding up the image searching process. When the image mosaic generating process starts, the original image is first divided into many tiles. For each tile, its colour feature is extracted and represented in the same way as all the DB images. Its binary signatures are used to match metadata of all DB images to select a set of candidate images. Then, based on two different approaches we proposed, the best matching image for this tile is selected. The final image mosaic is generated by substituting each tile with its best matching image. There are many parameters used in our system to control the generating process, thus affecting the processing time as well as the quality of image mosaics. These parameters include the number of quantization colours, the size of the image database, the scheme to extract images' LCHs, the size of the candidate image list, and the approach used to select the best matching image for a tile.

Our experimental results show that each of 5 parameters plays a different role in controlling the image mosaic generating process. The number of quantization colours used in our system is shown to be the most important parameter. It dramatically affects both the processing time and the quality of an image mosaic. And 64 quantization colours is shown to be the best choice among all the values we tested. With respect to the processing time, the size of the candidate list and image database also play important roles. However, neither the LCH extracting scheme nor the best matching image selecting approach affects an image mosaic generating process a great deal. With respect to the quality of an image mosaic, the size of the image database and the best matching image selection approach are two parameters affecting the process. In addition, considering the processing time and image mosaic's quality together, the best value or approach for each parameter is selected by us. Using these parameters with their best values, our system can generate very good image mosaics similar to the original image in acceptable time.

Another contribution of this thesis is an image mosaic evaluating measure we proposed, which is called the MLCH distance measure. This distance measure is based on human perceptions of images mosaics, which uses the increasing speed of LCH distance as the measure to evaluate an image mosaic's quality. The experimental results show that our proposed measure is the most stable one we tested, and outperforms the GCH distance measure.

5.2 Future Work

Our thesis focuses on the design and implementation of an image mosaic generating system. Some parameters are also experimented with to test their effects on the processing time and image mosaic's quality. Though, there are also some other challenging areas which may be investigated.

- During colour feature extracting, RGB colour space is used in our system.

Other colour spaces could also be used such as L*a*b and L*u*v.

- Non-uniform colour quantization approaches could be used instead of the uniform approach used in our system.
- Besides 4-block and 5-block LCH extracting schemes, there are many other schemes to extract LCHs of an image. They also can be investigated.
- Different approaches to select the best matching image for an input tile exist. More work can be done on this topic.
- Investigation on considering different tiles to have the same best matching image can be performed so that more processing time can be saved.

Bibliography

[1] Mosaic magic v2.0. http://www.fishsoft.co.uk/mosaic_magic.shtml.

[2] A. D. Bimbo. *Visual Information Retrieval.* Morgan Kaufmann Publishers, San Francisco, California, 1999.

[3] C. C. Chang and J. H. Jiang. A fast spatial retrieval using a superimposed coding technique. In *Proceedings of the International Symposium on Advanced Database Technologies and Their Integration*, pages 71–78, Nera, Japan, 1994.

[4] V. Chitkara. Color-based image retrieval using compact binary signatures. Master's thesis, Computing Science Department, University of Alberta, 2001.

[5] C. Colombo, A. Rizzi, and I. Genovesi. Histogram families for colour-based retrieval in image databases. In *Proceedings 9th International Conference on Image Analysis and Processing ICIAP'97*, pages (II)204–211, Firenze, Italy, September 1997.

[6] R. Echen. Art from text. http://www.rechen.f2s.com/arttext.html.

[7] E. A. El-Kwae and M. R. Kabuka. A robust framework for content-based retrieval by spatial similarity in image databases. *ACM Transactions on Information Systems (TOIS)*, 17:174–198, April 1999.

[8] F. Ennesser and G. Medioni. Finding waldo, or focus of attention using local color information. *I.E.E.E. Transactions on Pattern Analysis and Machine Intelligence*, 17:8, August 1995.

[9] P.G.B. Enser. Progress in documentation: Pictorial information retrieval. *Journal of Documentation*, 51:126–170, 1995.

[10] A. Finkelstein and M. Range. Image mosaics. Technical Report TR-574-98, Computer Science Department, Princeton University, 1998.

[11] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image content and video content: The QBIC system. *IEEE Computer*, pages 23–32, September 1995.

[12] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison Wesley Publishing company, 1993.

[13] M. Hartmann. 10000 zombies. http://www.xoz.net/vs/014.html.

[14] W. L. Hunt. The phototiled pictures. http://home.earthlink.net/ wl-hunt/index.html.

[15] P. M. Kelly, T. M. Cannon, and D. R. Hush. Query by image example: the candid approach. *Storage and Retrieval for Image and Video DatabasesIII, SPIE*, 2420:238–248, 1995.

[16] S. Y. Lee and M. K. Shan. Access methods of image databases. *International Journal of Pattern Recognition and Aritifical Intelligence*, 4:27–44, 1990.

[17] W. Y. Ma and B. S. Manjunath. Netra: A toolbox for navigating large image databases. *Multimedia systems*, 7:184–198, 1999.

[18] B. M. Mehtre, M. Kankanhalli, and W. F. Lee. Shape measures for content based image retrieval: A comparison. *Information Processing and Management*, 33:319–337, 1997.

[19] J. Miller, G. Pass, and R. Zabih. Comparing images using colour coherence vectors. In *Proceedings of the 4th ACM International Conference on Multimedia*, pages 65–73, Boston, Massachusetts, USA, November 1996.

[20] S. Mukherjea, K. Hirata, and Y. Hara. Amore: A world wide web image retrieval engine. *The WWW Journal*, 2:115–132, 1999.

[21] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glassman, D. Petkovic, and P. Yanker. The QBIC project: Querying image by content using colour, texture and shape. *SPIE*, 1908:115–132, February 1993.

[22] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18:233–254, 1996.

[23] Y. Rubner and C. Tomasi. Texture-based image retrieval without segmentation. In *Proceedings of the International Conference on Computer Vision, IEEE*, pages 1018–1024, Kerkyra, Greece, September 1999.

[24] Y. Rui, T. S. Huang, and S. F. Chang. Image retrieval: Past present and future. *Journal of Visual Communication and Image Representation*, 10:1–23, 1999.

[25] Y. Rui, A. She, and T. Huang. Modified fourier descriptors for shape representation - a practical approach. In *Proceedings of the First International Workshop on Image Databases and Multi Media Search*, pages 22–23, Amsterdam, Netherlands, August 1996.

[26] S. Sclaroff, L. Taycher, and M. LaCascia. Imagerover: A content-based image browser for the world wide web. Technical Report 1997-005, June 1997. http://citeseer.nj.nec.com/sclaroff97imagerover.html.

[27] R. Silvers. Digital composition of a mosaic image, October 2000. US Patent and Trademark Office. http://www.uspto.gov.

[28] J. Smith and S. Chang. In M. T. Maybury, editor, *Intelligent Multimedia Information Retrieval*, chapter Querying by color regions using the VisualSEEk content-based visual query system. 1996.

[29] J. Smith and S. Chang. Automated binary texture feature sets for image retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2239–2242, Atlanta, GA, USA, 1996.

[30] J. R. Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Compression and Analysis.* Ph.d. thesis, Graduate School of Arts and Science, Columbia University, February 1997.

[31] J. R. Smith and S. F. Chang. Tools and techniques for color image retrieval. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 426–437, 1996.

[32] R. O. Stehling, M. A. Nascimento, and A. X. Falcao. On shapes of colours for content-based image retrieval. *International Workshop on Multimedia Information Retrieval*, November 2000.

[33] Al Stevens. Review of the book photomosaics. *Electronic Review of Computer Books*, 1997.

[34] M. Stricker and M. Orengo. Similarity of colour images. In *Proceedings of SPIE Conference, Storage Retrieval for Still Images Video Databases*, volume 2420, pages 381–392, San Jose, CA, September 1995.

[35] M. J. Swain and D. H. Ballard. Colour indexing. *International Journal of computer vision*, 7:11–32, 1991.

[36] K. L. Tan, B. C. Ooi, and C. Y. Yee. An evaluation of colour-spatial retrieval techniques for large image database. In *Multimedia Tools and Applications.* Kluwer Academic Publishers, 2001.

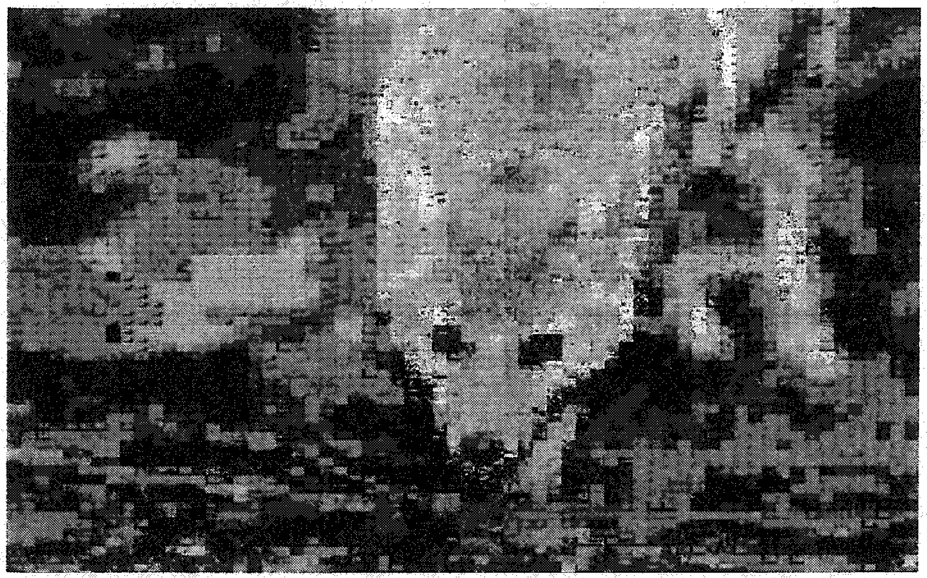
[37] N. Tran. Generating photomosaics: An empirical study. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 105–109, San Antonio, Texas, USA, 1999.

[38] L. Yang and F. Albregtsen. Fast computation of invariant geometric moments: A new method giving correct results. In *Proceeding of the Interna-*

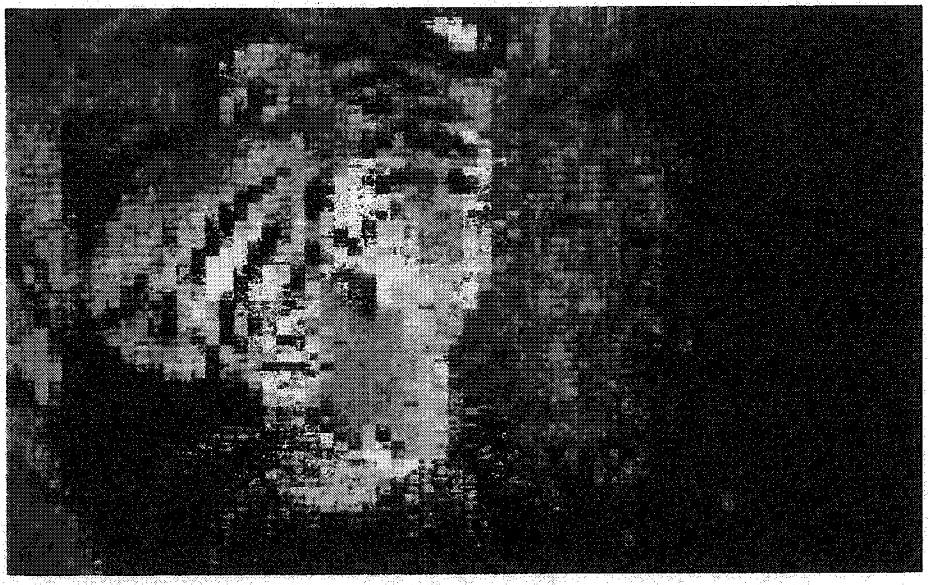
tional Conference on Pattern Recognition (ICPR) 94, pages 201–204, San Antonio, Texas, USA, 1994.

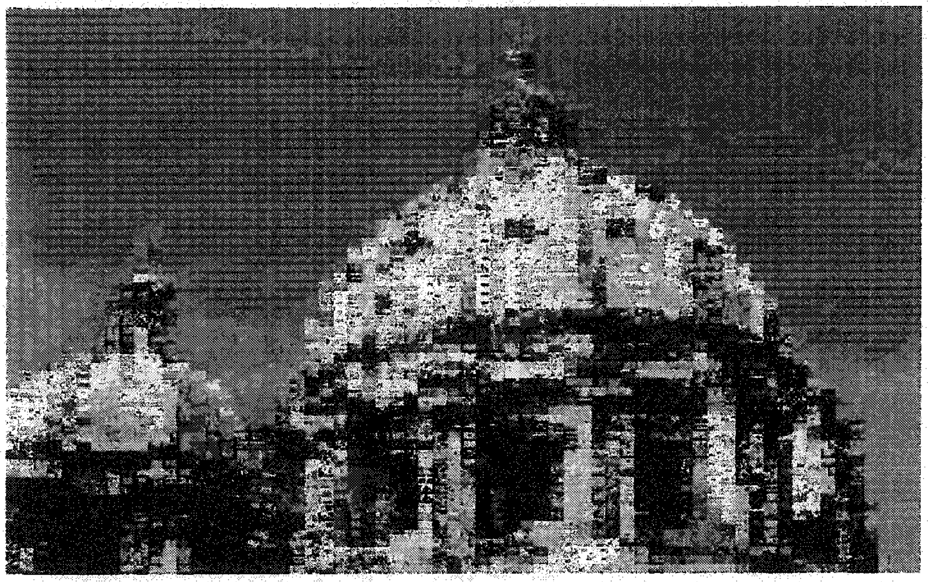
Appendix A

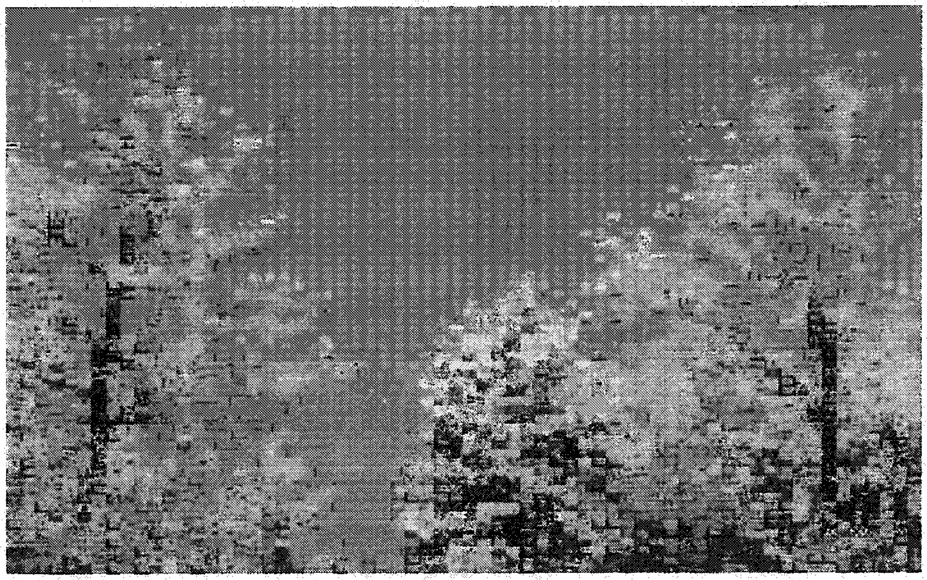
Image mosaics generated using Mosaicture

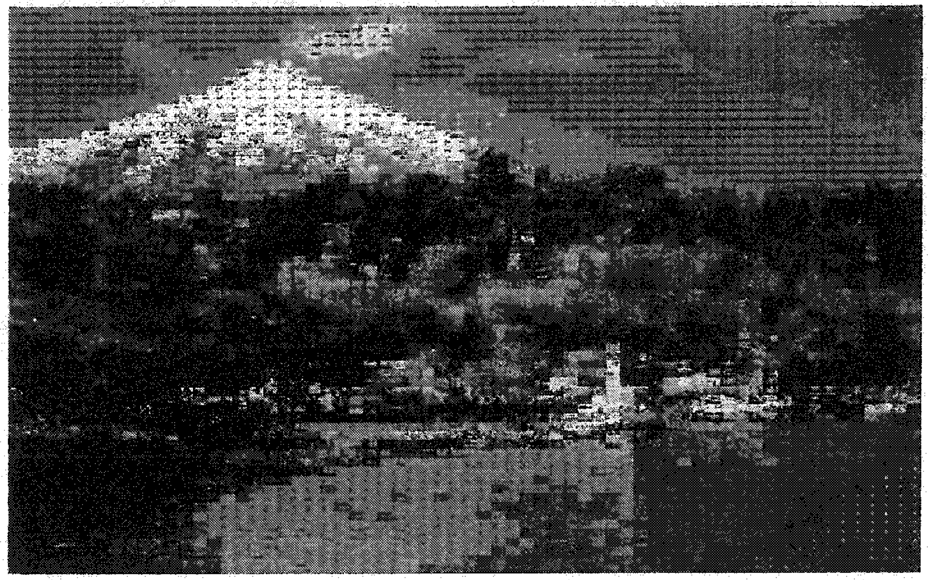


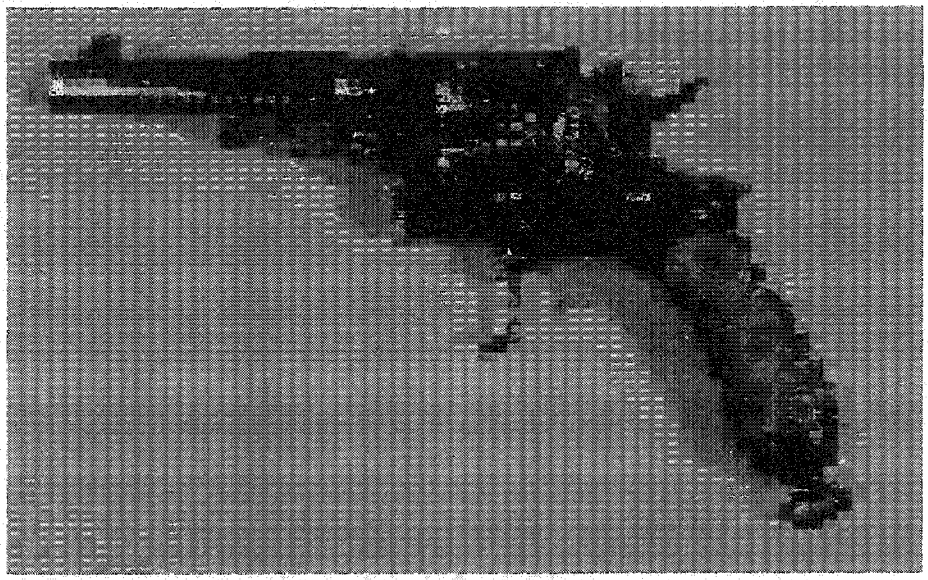
[0]Default values are used for all parameters when generating these image mosaics











Appendix B

Image mosaics generated using different parameters

Parameter Name	Abbreviation
The size of image DB	D
Number of quantization colours	Q
The size of candidate list	L
LCH extraction scheme	E
The best matching image selection scheme	B

Table B.1: The abbreviations of parameters used in following examples

[0]A large number of image mosaics generated using different parameters can be found at http://www.cs.ualberta.ca/~zach/thesis

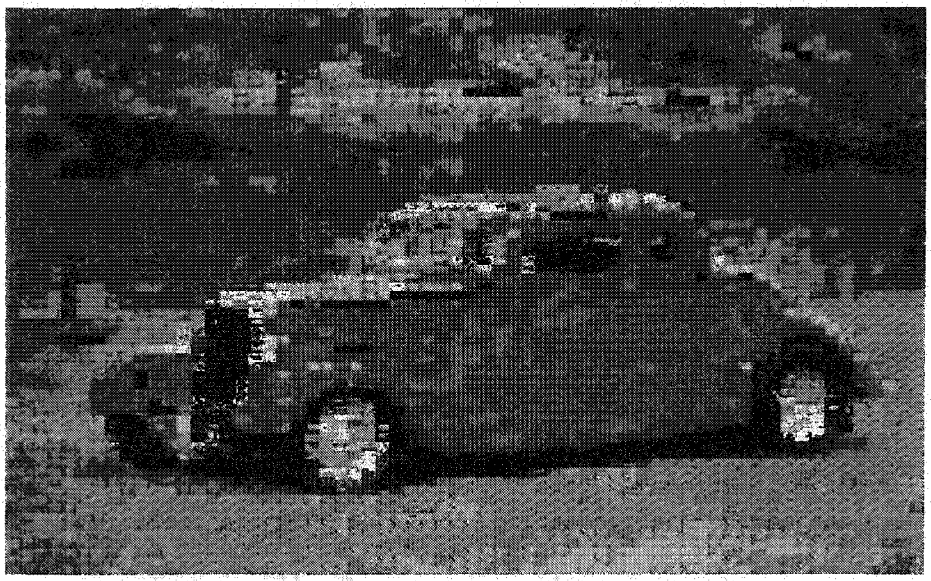


Figure B.1: D=33856, Q=64, L=30, E=4-block, B=MOSS

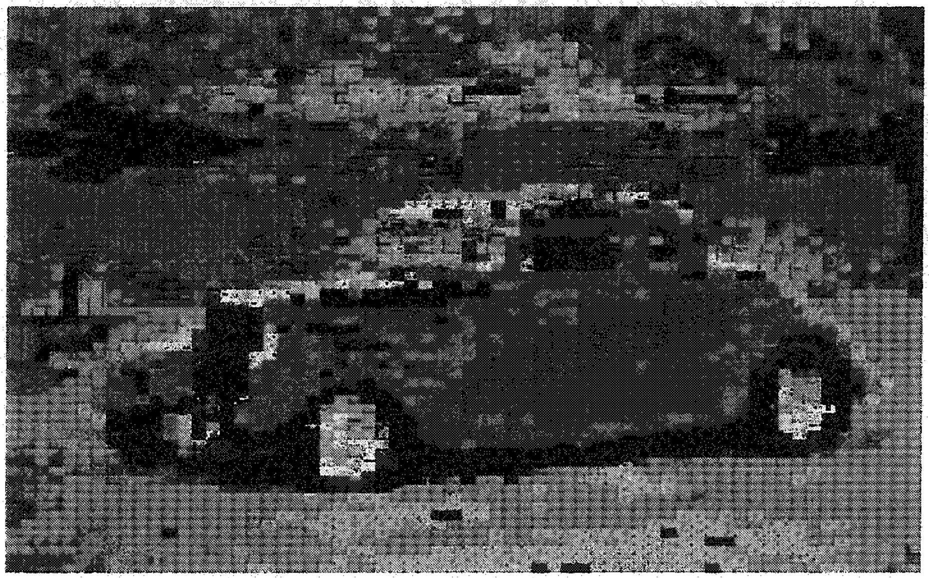


Figure B.2: D=13846, Q=64, L=30, E=4-block, B=MOSS

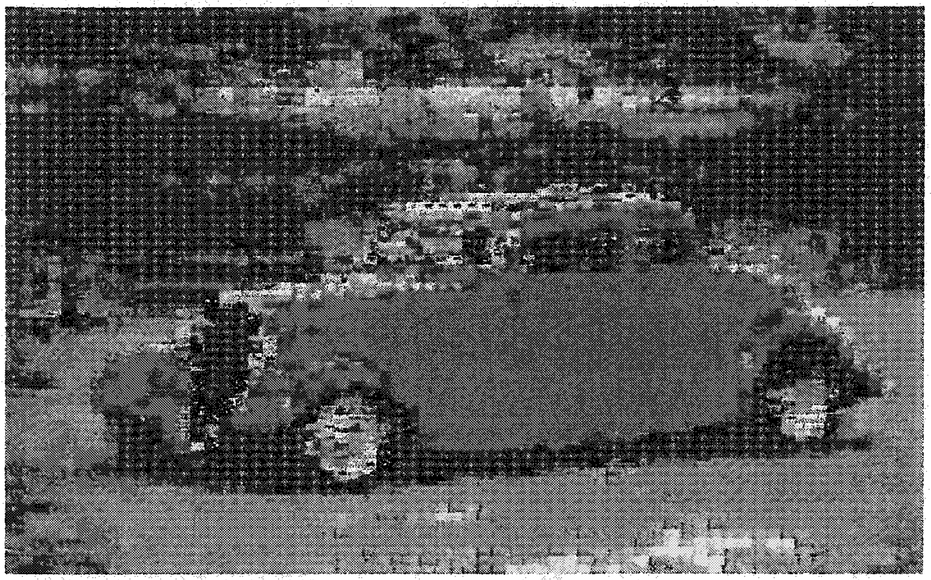


Figure B.3: D=33856, Q=8, L=30, E=4-block, B=MOSS

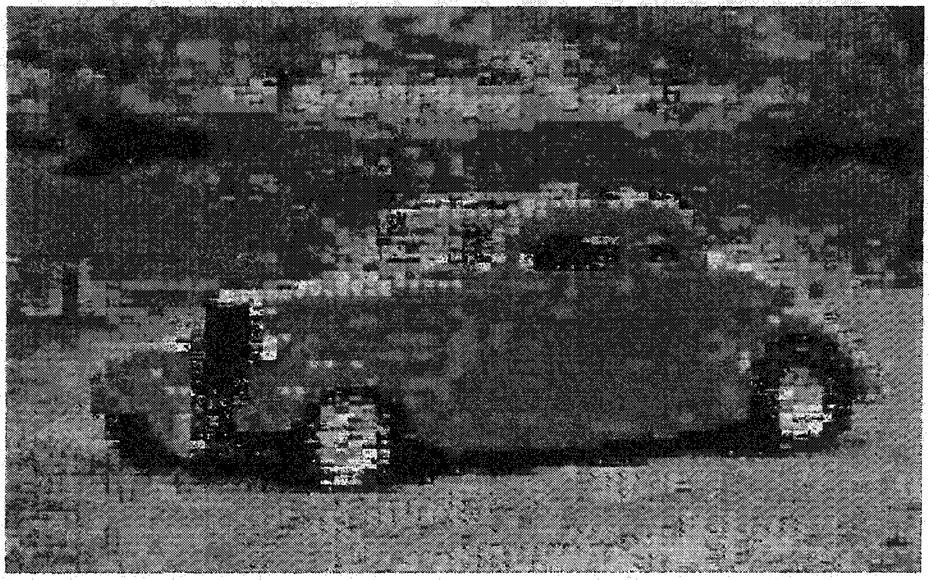


Figure B.4: D=33856, Q=64, L=3, E=4-block, B=MOSS

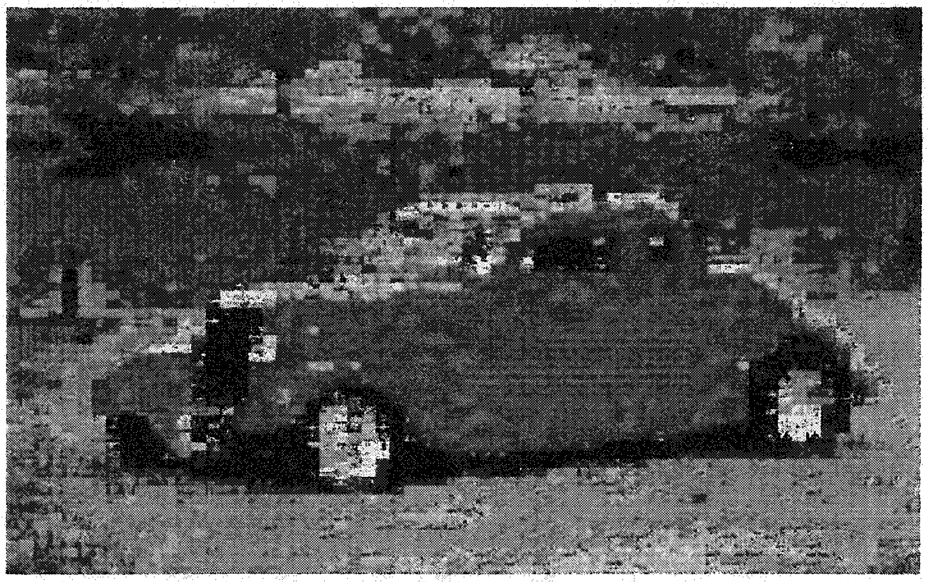


Figure B.5: D=33856, Q=64, L=30, E=5-block, B=MOSS

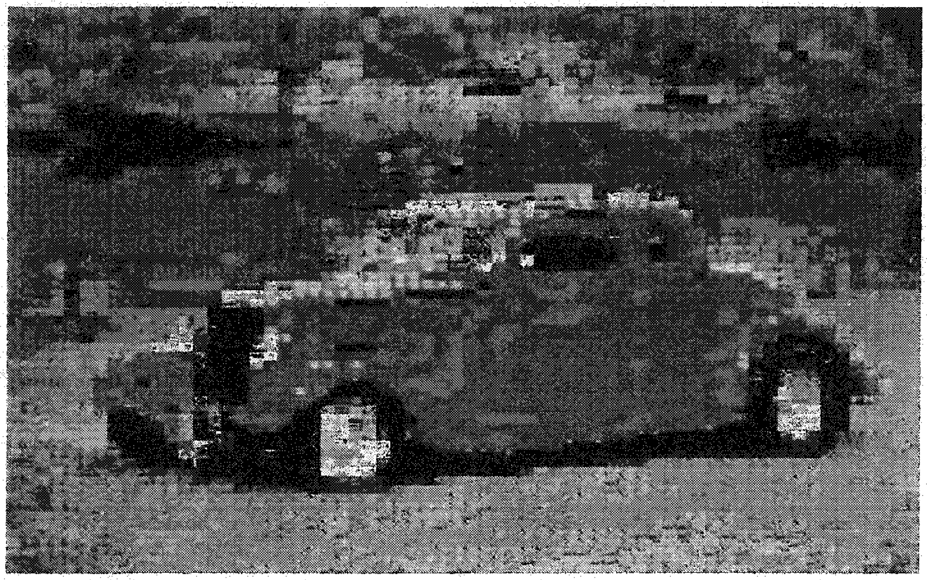


Figure B.6: D=33856, Q=64, L=30, E=4-block, B=DOSS

Appendix C

Detailed look at some example image mosaics