University of Alberta

A Robust CBIR Approach Using Local Color Histograms

by

Shengjiu Wang

Technical Report TR 01-13
October 2001

DEPARTMENT OF COMPUTING SCIENCE
University of Alberta
Edmonton, Alberta, Canada

A Robust CBIR Approach Using Local Color Histograms

Shengjiu Wang

Abstract

Global color histograms are well-known as a simple and often way to perform color-based image retrieval. However, it lacks spatial information about the image colors. The use of a grid of cells superimposed on the images and the use of local color histograms for each such cell improves retrieval in the sense that some notion of color location is taken into account. In such an approach however, retrieval becomes sensitive to image rotation and translation. In this thesis we present a new way to model image similarity, also using colors and a superimposing grid, via bipartite graphs. As a result, the technique is able to take advantage of color location but is not sensitive to rotation and translation. Experimental results have shown the approach to be very effective. If one uses global color histograms as a filter then our approach, named Harbin, becomes quite effcient as well (i.e., it imposes very little overhead over the use of global color histograms).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last few years, the rapid growth of the Internet has enormously increased the number of image collections available. The accumulation of these image collections (including art works, satellite and medical imagery) is attracting more and more users in various professional fields—for example, geography, medicine, architecture, advertising, design, fashion, and publishing. Meanwhile, the study of image retrieval, which is concerned with effectively and efficiently accessing desired images from large and varied image collections, has become more interesting and more challenging.

Image retrieval is concerned with techniques for storing and retrieving images both efficiently and effectively. Early image retrieval methods locate the desired images by matching keywords that are assigned to each image manually [FMBR95]. However, as a result of the large number of images in collections, manual processing has become impractical. As well, because we are unlikely to foresee all the query keywords that will be used in a retrieval process, it is impractical to assign keywords to every image, so the effectiveness of classic image retrieval is very limited.

Content-based image retrieval (referred to as CBIR in this thesis), which is based on automatically-extracted primitive features such as color, shape, texture, and even the spatial relationships among objects, has been employed since the 1990's. In the last ten years, a great deal of research work on image retrieval has concentrated on CBIR technology. Some commercial products based on CBIR technology have come to the marketplace, well-known examples including QBIC [NBE+93] and Virage [GJ97].

As a communications medium, images contain semantic information (for example images containing a happy dog). Such semantic features may be necessary for some queries, called semantic-level queries, but existing CBIR techniques cannot extract the semantic features effectively [GR95], although some semantic-level queries can be represented by primitive feature queries. For example, a semantic-level query that searches for images of a blue sky can be interpreted by a primitive feature query, which retrieves images that have the color blue on the top. The capability of present CBIR systems has been limited by their use of only primitive features, so they cannot satisfy most semantic-level query demands.

Although CBIR has the above inherent limitation, its practical results still provide us with some positive feedback, and further research in this area will lead to much better results.

1.1 Primitive features of images

Primitive features denote some general visual characteristic including color, shape, texture, and spatial relationships among objects, and these features can be used in most CBIR applications. The color feature, which is widely used in CBIR systems, will be discussed in detail in Chapter 2.

Figure 1.1(a) gives a sample image containing a flower, with the thick white edges outlining the shape of the flower. We can then search for images of flowers that have shapes similar to the flower in our sample image. The shape of an object or region captures very important content information in an image [Bie87]. The retrieval by shape always requires robust edge detection and image segmentation algorithms [GK94] [LZC+94] [HH94] to detect object or region boundaries. In general, the shape analysis algorithms can be divided into two classes: one is based on the representation of the outer boundaries of objects, such as Fourier Descriptor [RSH96]; the other is based on the representation of entire shape regions, such as Moment Invariants [YA94]. In addition, matching the shapes of three-dimensional objects is more challenging than working with the two-dimensional representations. Some work has already been done in this area [CS96] [DPS98].

Figure 1.1(b) shows the texture of snakeskin: we can search for images of snakes by looking for those containing similar texture. Texture features refers to the visual patterns of regions of images, therefore many techniques in Pattern Recognition and Computer Vision can be used in Retrieval by Texture. Since images with similar color intensity can have very different textures, the texture feature can often be used to distinguish between images with similar color distribution, such as sky and sea, or leaves and grass. Measures to compute texture similarity based on visually meaningful texture properties [TMY78] [LP96] that include contrast, coarseness, directionality, regularity, periodicity, and randomness, were used in current CBIR systems such as the QBIC system [NBE+93] and MARS system [HMR96]. Other techniques for retrieval by texture include Wavelet transform [MM96] and fractals [KMN98].

Spatial relationships between objects or regions can often contain relevant information for image retrieval. In general, the representation structures for spatial relationships can be divided into two groups: objected-based and relation-based. The object-based structures retrieve images by matching positions of objects within images but do not explicitly store relationships. The representative structures include Grid file [NHS84], and Quad-tree [Sam84], as well as R-tree and its families [GG98]. The relation-based structures symbolize objects within images and preserve a set of spatial relationships. The relation-based structure can be further divided into different models, such as symbolic projection-based models [CL84] [CSY86] [CJ90], transformation invariant models [PSTT93] [PO93], logic-based models [BVZ95], and point-set models [Ege91]. Figure 1.1(c) shows how to use 2D-strings [CL84], a symbolic projection-based model, to represent the spatial relationships among the objects in the sample image. In the two strings, operator ”¿” means ordering and operator ”=” means overlapping; the ”u string” shows the spatial relationships along the u axis, and the ”v string” shows the spatial relationships along the v axis. We can then search for similar images by searching for two similar strings.

Many technologies based on these primitive features have been proposed in the past decade. Each of them is suitable for some specific queries; at the same time, all these technologies have limitations. For example, when searching for airplanes using a sample image of a white airplane, retrieval by shape may return good results. If we use retrieval by color, since the answer set may include a variety of images containing white or silver elements, the results may be very poor. On the other hand, by combining these features, CBIR systems can tackle complicated queries. For example, the query ”search for images containing a red ball” can be presented by combining the shape feature ”circle” and the color feature ”red”. However, it is very difficult to find a general way to query an image using a combination of these primitive features. As the building blocks of CBIR systems, the technologies to achieve retrieval by color, retrieval by shape, and retrieval by texture are still being widely studied by current researchers [RHC99] [Bim99] [Lu99].

1.2 Requirements for CBIR systems

We believe that the following four elements are important for building a CBIR system:

Techniques to automatically extract metadata containing primitive features of images. The metadata depict images with primitive features. Each primitive feature has its own representation format, such as the color histogram, which was first introduced in [SH91] and has been widely used to represent the color feature. Another example is the shape feature, which can be represented by sets of consecutive boundary segments [SSS00]. With

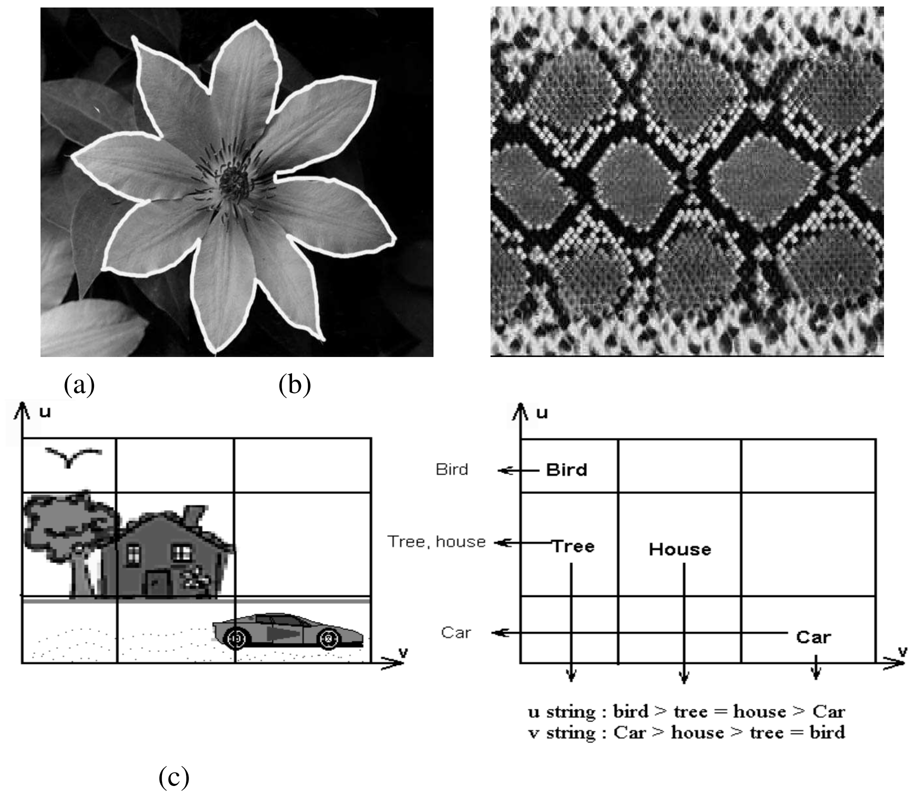


Figure 1.1: (a) the shape of a flower; (b) the texture of snakeskin; (c) spatial relationships

the appropriate metadata, current CBIR systems can achieve retrieval by color, by texture, by shape, by spatial relationship, and by a combination of the above [GR95].

Interfaces to obtain the users' query demands. As in any retrieval system, the retrieval process begins with a query requirement. Hence, it is crucial to capture the users' query demands accurately and easily. Query by text has been widely used in retrieval systems—for example, searching for desired books with keywords in a library. For current CBIR systems, the image retrieval process is usually carried out through an example image provided by the user, called query-by-example (referred to as QBE in this thesis). However, users cannot always submit an example image to the retrieval system. Typically, current CBIR systems solve this problem by offering an interface to specify or choose some primitive features for providing an example image. For instance, when using IBM's QBIC system [NBE+93], users can specify the color feature query by choosing relative amounts of red, green, and blue, or by selecting a desired color from a color palette. Also, users can choose the desired texture for the texture feature query, and draw a sketch for the shape feature query. Sometimes, when querying by an example image, we are not interested in the whole image but just some parts or objects. Also, we may not be concerned about the location of objects. For instance, in the case of an image of a green mountain with a sunrise, we might want to retrieve images with only a sunrise. We do not care whether the sunrise is over a green mountain or a blue sea, and we do not care whether the sunrise is on the left top of the image or in the right middle of the image. In other words, the process involves sub-image or object retrieval. Given an example image, we choose a significant part, and the CBIR system searches for the desired images that contain the part similar to the part in the sample image.

Methods to compare the similarities (or differences) between images. CBIR systems require methods that are based on primitive features to compare the similarities or differences between an example image and all the images in the image collection. However, the similarities or differences between images cannot be quantified in an ideal manner. The extent to which the images are similar will change when query requirements are varied. For instance, in the case of two pictures, one of a blue sea with a sunrise and the other of a green mountain with a sunrise, when the sunrise is considered, the similarity between these two images should be high, but if the object of interest is the blue sea, the similarity between these two images should be low. We believe that it would be very difficult to find a method to measure the similarities or differences between images accurately for all kinds of query demands. In other words, every retrieval method will have its own limits. For example, it will be hard for a color-based image retrieval technology to differentiate an image of blue sky from an image of blue sea. Therefore, when evaluating a CBIR technology, one should remember that the retrieval effectiveness of that technology depends on the types of query requirements that users make. As far as we know, there are no benchmarks that contain a set of images on which CBIR technologies can be tested, nor there are representative queries and their expected answers set to evaluate the retrieval effectiveness of CBIR technologies. When we use a precision-recall graph or some coefficients as assessment methods to evaluate the retrieval effectiveness of CBIR technologies, a topic which will be discussed later in this thesis, we usually need a set of representative queries and their corresponding answer sets that contain all and only the desired images. In practice, current CBIR systems commonly use QBE. To that effect we have used an ad-hoc data set (also discussed later in the thesis).

Techniques to efficiently index and store metadata. For a large image collection, the storage space required for the metadata is usually non-trivial. A CBIR system must possess efficient techniques to compress the metadata. Also, retrieving images and other visual resources, such as video, demands that standards be set to describe the metadata. The MPEG-7 standard[1] is becoming the most important standard to describe all kinds of metadata for both images and video data.

When a query is processed against a large image database, it is often unacceptable to compare the similarity between the query image and all the images, one by one. Because users only need images having high similarity to the example image, index structures, which can help to prevent sequential searches and improve query efficiency, should be used in CBIR systems. Further, for frequently varied image databases, dynamic index structures are necessary. When the content of images is represented by low-dimension vectors and the distance between images is defined (possibly as a spatial distance calculated with the Euclidean distance), the R-tree and its families [GG98] can be used to index images dynamically. When the distance is not defined as the vector space, or when the vector space is highly dimensional, or when what we have is only a distance function that can define a metric space, methods to index images based on the distance function in the metric space are available. When a distance function can satisfy the metric properties – nonnegativity, symmetry, and triangle inequality – M-trees [CPZ97], which are dynamically balanced trees, can also be applied to index objects.

In this thesis we will propose an effective method to compare similarities between images, and its index structure will be studied in the future.

1.3 CBIR systems

In this section, we will select several existing CBIR systems and describe their characteristics.

IBM's QBIC system [NBE$^+$93] is the first commercial CBIR system and probably the best known of all CBIR systems. QBIC supports users to retrieve images by color, shape, and texture. QBIC provides several query methods: Simple, Multi-feature, and Multi-pass. In the Simple method, a query is processed using only one feature. A Multi-feature query involves more than one feature, and all features have equal weights during the search. A Multi-pass query uses the output of a previous query as the basis for further refinements. Users can draw and specify color and texture patterns in desired images. In QBIC, the color similarity is computed

[1] `http://drogo.cselt.stet.it/mpeg/standards/mpeg-7/mpeg-7.html`

by a quadratic metric using k-element color histograms [FBF+94], and the average colors are used as filters to improve query efficiency. Its shape function retrieves images by shape area, circularity, eccentricity, and major axis orientation. Its texture function retrieve images by global coarseness, contrast, and directionality features. An R*-tree [BKSS90] is used as the underlying indexing structure to index multi-dimensional features. The online demo of QBIC is at: `wwwqbic.almaden.ibm.com`.

The Photobook system [PPS94], developed at the Massachusetts Institute of Technology, allows users to retrieve images by color, shape and texture features. This system provides a set of matching algorithms, including Euclidean, mahalanobis, divergence, vector space angle, histogram, Fourier peak, and wavelet tree distances, as distance metrics. In its most recent version, users can define their own matching algorithms. The system includes a distinct interactive learning agent (FourEyes) [MP96], which is a semi-automated tool and can generate query models based on example images provided by users. This allows users to directly address their query demands for different domains and, for each domain, users can obtain an optimal query model. The online demo can be seen at: `http://www-white.media.mit.edu/vismod/demos/photobook`.

Both VisualSEEK [SC96b] and WebSEEK [SC97] were developed at Columbia University. The VisualSEEK system is an image database system; It allows users to retrieve images using color, spatial layout, and texture features. Color Set and the Wavelet Transform based texture [SC95] are used to represent these features. In addition, VisualSEEK allows users to form queries by specifying color regions and their spatial locations. WebSEEK is an image catalog and search tool for the web. This system provides a prototype to catalog images and videos on the web using a combination of the text-based processing and content-based visual analysis. The online demo can be seen at: `http://disney.ctr.columbia.edu/webseek.`

RetrievalWare [Dow93], developed by Excalibur Technologies Corp., lets users retrieve images by Color Content, Shape Content, Texture Content, Brightness, Color Structure, and Aspect Ratio. Users can adjust the weights of these features during the search process. The online demo can be seen at: `http://vrw.excalib.com:801`

The IMatch system allows users to retrieve images by color, texture, and shape. IMatch supports several query methods to query similar images: Color Similarity, Color and Shape (Quick), Color and Shape (Fuzzy), and Color Distribution. Color Similarity queries for images similar to an example image based on the global color distribution. Color and Shape (Quick) queries similar images for a given image by combining shapes, textures, and colors. Color and Shape (Fuzzy) performs additional steps to identify objects in example images. Color Distribution allows users to draw color distributions, or specify the overall percentage of one color in desired images. IMatch also supports non-CBIR features to identify images: binary identical images, duplicate images that have been resized, cropped, or saved in different file formats, and images that have similar file names to the given images. Demos for IMatch can be downloaded from: `www.mwlabs.de/download.htm.`

1.4 Summary of contributions

In this thesis, we will examine only color-based image retrieval techniques for comparing similarities between images. We propse a new method (referred to as Harbin in this thesis) based on color features. Two traditional color-based image retrieval techniques (GCH and LCH) are the basis of Harbin. By constructing weighted bipartite graphs, our approach reduces the problem of calculating the distance between images to the problem of finding the minimum or maximum cost matching in the bipartite graph; the cost of the matching is then treated as the distance or similarity between the corresponding images.

We will perform a variety of experiments on our CBIR database, and compare Harbin with two traditional color-based image retrieval techniques. As well, through experiments, we are able to know how to set parameters in Harbin. The results demonstrate that Harbin is more effective than GCH and LCH. Given that the process of finding matchings in bipartite graphs may be computationally expensive, we will show that GCH can be used as a quick filter for Harbin. Hence, Harbin is also fairly efficient, even though we do not address the issue of indexing in this thesis, since the main contribution of this thesis is a new effective approach for CBIR.

The thesis is divided into five chapters. Chapter 2 describes some work related to color-based image retrieval. In particular, we describe two traditional color-based image retrieval approaches—Global Color Histograms and Local Color Histograms—which are the starting points of our approach. Methods for quantifying the color space, extracting and normalizing color histograms, and calculating the distance between color histograms are also outlined in this Chapter.

In Chapter 3, we present our contribution, the Harbin approach. We explain the motivation behind this approach, and describe how it works. The problem of weighted matching on bipartite graphs and related algorithms is described as well.

In Chapter 4, we evaluate our method by means of extensive experiments. Our method contains three main parameters: (1) the number of colors in color histograms, (2) the number of blocks used to segment images, and (3) Δ, by which we can decide whether to keep an edge in bipartite graphs.

In the last chapter, we present our conclusion, summarize the contributions as well limitations of our method, and suggest directions for future work.

Chapter 2

The Color Feature

The human eye is sensitive to colors, and color features are one of the most important elements enabling humans to recognize images. Color features are, therefore, fundamental characteristics of the content of images. Color features can sometimes provide powerful information for categorizing images, and they are very useful for image retrieval. Therefore, color-based image retrieval is widely used in CBIR systems.

Color histograms are generally used to represent the color features of images [NBE+93] [PZM96] [SH91]. Before using color histograms, however, we need to select and quantify a color space model and choose a distance metric. We now address those particular issues.

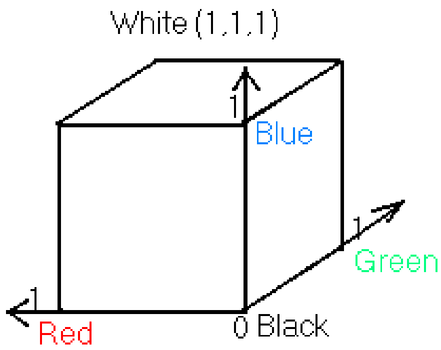
2.1 Color Space

Colors are commonly defined in three-dimensional color spaces. The color space models [GW92] can be differentiated as hardware-oriented and user-oriented. The hardware-oriented color spaces, including RGB, CMY, and YIQ, are based on the three-color stimuli theory. The user-oriented color spaces, including HLS, HCV, HSV HSB, MTM, CIE-LAB, and CIE-LUV, are based on the three human percepts of colors, i.e., hue, saturation, and brightness. The color space model can also be distignuished as uniform and non-uniform depending on differences in color space as perceived by humans (in fact, there is no truly uniform color space). The approximate uniform color spaces include MTM, CIE-LAB, and CIE-LUV.

Commonly used image formats, such as JPEGs, GIFs, and BMPs always store and show colors in RGB color space. Image retrieval based on RGB color space, therefore, will not need color space transformation, and is consequently facilitated. However, because the RGB color space has the deficiency of not being perceptually uniform, in color based image retrieval, the RGB color space may be transformed into other spaces that improve the perceptual uniformity. In practice, a variety of color spaces have been used: the Munsell color space in the QBIC system [NBE+93], the CIE-LUV color space by Gray [Gra95], the HSV color space by Smith [Smi97], and the RGB color space by Pass et al [PZM96]. There is no evidence, however, to show which color space always generates the best retrieval results. Since our techniques are independent of color spaces, in this thesis we will study all the techniques using the RGB color space.

The RGB color space (see Figure 2.1.(a)) is defined as a unit cube with red, green, and blue axes; hence, a color in an RGB color space is represented by a vector with three coordinates. When all three values are set to 0, the corresponding color is black. When all three values are set to 1, the corresponding color is white.

In the color space quantization, we reduce the color space of all possible colors to a set of discrete colors. In fact, this process of quantization is the same as the process of reducing colors. We divide the large cube into sub-cubes, and each sub-cube can be represented by a single color. For example (see Figure 2.1.(b)), we divide the large cube into 64 sub-cubes (4^3) by separating the red, green, and blue axes into 4 sub-divisions; all the possible colors located in a sub-cube will be represented by a single color.



(a) (b)

Figure 2.1: RGB Color Space

For current computer systems, the RGB color space is usually represented by a 24-bit true color system. In a 24-bit color system, a color is specified by three integers: {red, green, and blue}, and these three integers range from 0 to 2^8-1 respectively, so giving 16,777,216 (2^{24}) colors in total. Because the process of quantizing the RGB color space is the same as for the process of reducing colors, we simply quantize the color space by reducing the number of colors from 24-bit colors to n^3 colors, as demonstrated by the following

When reducing a color $\{r, g, b\}$ with 24-bit colors to a new color $\{r', g', b'\}$ with n^3 colors, we set $r' = \frac{n \times r}{2^8}$, $g' = \frac{n \times g}{2^8}$, $b' = \frac{n \times b}{2^8}$. Therefore, after reducing the number of colors, we get $n \times n \times n = n^3$ colors.

2.2 Color histograms

In this thesis, we define color histograms as a set of bins where each bin denotes the probability of pixels in the image being of a particular color. A color histogram H for a given image is defined as a vector:

$$H = \{H[0],\ H[1],\ H[2],\ ...,\ H[i],\ ...,\ H[N]\},$$

where i represents a color in the color histogram and corresponds to a sub-cube in the RGB color space, $H[i]$ is the number of pixels in color i in that image, and N is the number of bins in the color histogram, i.e., the number of colors in the adopted color model.

Typically, each pixel in an image will be assigned to a bin of a color histogram of that image, so for the color histogram of an image, the value of each bin is the number of pixels that has the same corresponding color. In order to compare images of different sizes, color histograms should be normalized. The normalized color histogram H' is defined as:

$$H' = \{H'[0],\ H'[1],\ H'[2],\ ...,\ H'[i],\ ...,\ H'[N]\},$$

where $H'[i] = \frac{H[i]}{P}$, P is the total number of pixels in an image (the remaining variables are defined as before).

An ideal color space quantization presumes that distinct colors should not be located in the same sub-cube and similar colors should be assigned to the same sub-cube. Using few colors will decrease the possibility that similar colors are assigned to different bins, but it increases the possibility that distinct colors are assigned to the same bins, and that the information content of the images will decrease by a greater degree as well. On the other hand, color histograms with a large number of bins will contain more information about the content of images, thus decreasing the possibility of distinct colors will be assigned to the same bins. However, they increase the possibility that similar colors will be assigned to different bins, the storage space of metadata, and the time for calculating the distance between color histograms. Therefore, there is a trade-off in determining how many bins should be used in color histograms. A typical figure found in the related literature is 64.

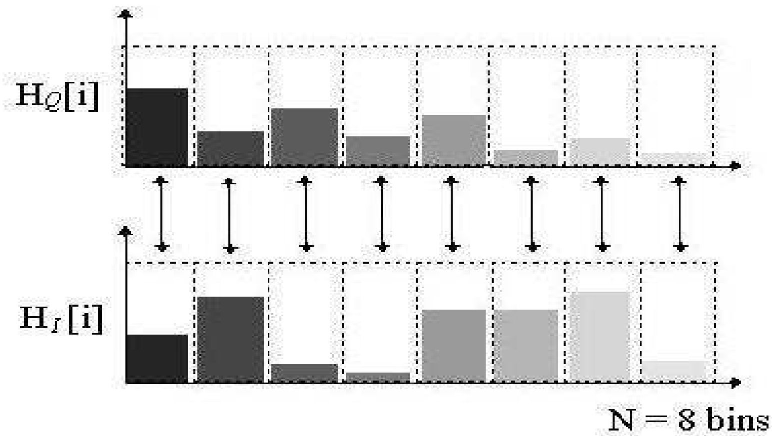


Figure 2.2: Minkoski-form distance metrics

2.3 Color Histogram Distance Metrics

A variety of metrics have been proposed to calculate the distance between color histograms. [Smi97] catalogued distance metrics into three classes, namely Minkowski-form distance, Quadratic-form distance, and Non-histogram distance.

2.3.1 Minkowski-form distance metrics

In [Smi97], Minkowski-form distance metrics compare only the same bins between color histograms (see Figure 2.2), and are defined as:

$$d(Q, I) = \sum_{i=1}^{N} |H_Q[i] - H_I[i]|^r,$$

where Q and I are two images, N is the number of bins in the color histogram (for each image we reduce the colors to N in the RGB color space, so each color histogram has N bins), $H_Q[i]$ is the value of bin i in color histogram H_Q, which represents the image Q, and $H_I[i]$ is the value of bin i in color histogram H_I, which represents the image I.

When $r = 1$, the Minkoski-form distance metric becomes L_1. When $r = 2$, the Minkoski-form distance metric becomes the Euclidean distance. In fact, this Euclidean distance can be treated as the spatial distance in a multi-dimensional space.

In this thesis, we will use the square root of Euclidean distance, which is defined as:

$$d(Q, I) = \sqrt{\sum_{i=1}^{N} (H_Q[i] - H_I[i])^2}$$

to calculate the distance between two color histograms.

2.3.2 Quadratic-form distance metrics

The QBIC project [NBE+93] used the Quadratic-form distance metric, which compares not only the same bins but multiple bins between color histograms (see Figure 2.3) and is defined as:

$$d(Q, I) = (H_Q - H_I)^t A (H_Q - H_I)$$

where Q and I are two images, H_Q is the color histogram of image Q, H_I is the color histogram of image I, $A = [a_{i,j}]$ is a $N \times N$ matrix, N is the number of bins in the color histograms, and $a_{i,j}$ denotes the similarity between colors i and j.

Quadratic-form distance metrics overcome a shortcoming of the Minkoski-form distance metrics in that the latter assumes that bins in color histograms are totally unrelated, while the former does not.

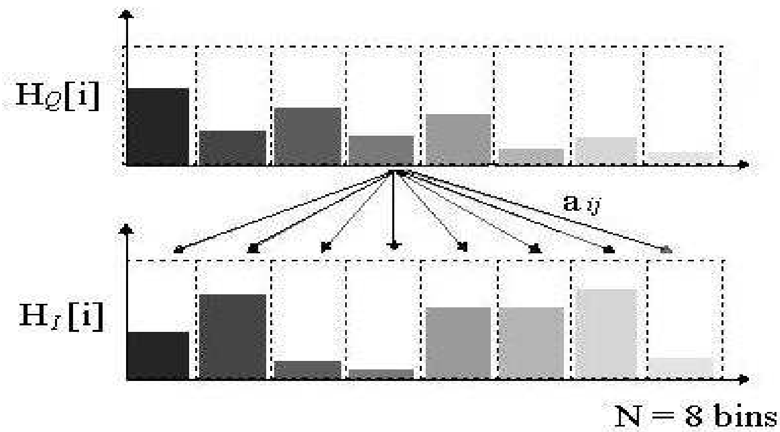


Figure 2.3: Quadratic-form distance metrics

2.3.3 Non-histogram distance metrics

Stricker and Orengo [SO95] proposed the Color Moments approach to overcome the quantization effect of the histograms. In this approach, color distribution features of images are represented by their dominant features (termed moments), namely average, variance, and skewness. The first moment is the average color of the image, the second is the standard deviation of each color channel, and the third is the third root of each color channel. They are defined as: $E_i = \frac{1}{F}\sum_{j=1}^{F} P_{ij}$, $\sigma_i = (\frac{1}{F}\sum_{j=1}^{F}(P_{ij}-E_i)^2)^{\frac{1}{2}}$, $s_i = (\frac{1}{F}\sum_{j=1}^{F}(P_{ij}-E_i)^3)^{\frac{1}{3}}$, respectively, where P_{ij} is the value of the i^{th} color channel at the j^{th} image pixel, E_i is the average color of the i^{th} color channel, σ_i is the standard deviation of the i^{th} color channel, s_i is the third root of the i^{th} color channel, and F is the total number of pixels. If Q and I are two images, and their color features are represented by r color channels, then the similarity between these two images is defined as:

$$d(Q,I) = \sum_{i=1}^{r}(W_{i1}|E_i^Q - E_i^I| + W_{i2}|\sigma_i^Q - \sigma_i^I| + W_{i3}|s_i^Q - s_i^I|)$$

where W_{i1}, W_{i2}, and W_{i3} are user specified weights.

Stricker and Orengo show that their approach produces slightly better results and runs faster than histogram-based methods (since each image will be represented by only 9 real numbers).

2.4 Traditional color-based techniques

There are two traditional techniques for color-based image retrieval: Global Color Histograms that represent images with single histograms; and Local Color Histograms that divide images into fixed blocks and, for each block, obtain its color histogram. Global Color Histograms do not capture the content of images adequately, whereas Local Color Histograms contain more information and also enable the color distances among regions between images to be compared. These techniques are suited to different types of queries, and we now discuss each of them in detail.

2.4.1 The Global Color Histogram

As we have discussed, the color histogram depicts color distribution using a set of bins. Using the Global Color Histogram (referred as to GCH in this thesis), an image will be encoded with its color histogram, and the distance between two images will be determined by the distance between their color histograms. For GCHs, we can use different metrics, which have been discussed previously (Section 2.3), to compute the distance between color histograms. The following example (see Figure 2.4) shows how a GCH works.

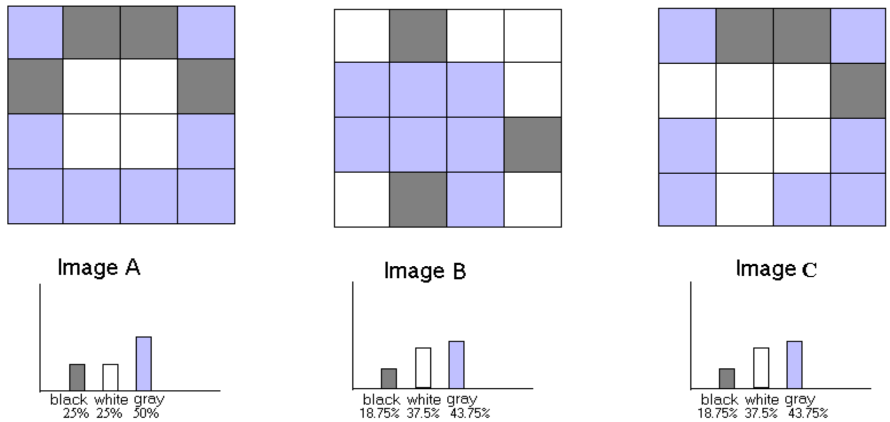


Figure 2.4: Three images and their color histograms

In the sample color histograms there are three bins: black, white, and gray. We note the color histogram of image A: $\{25\%, 25\%, 50\%\}$; the color histogram of image B: $\{18.75\%, 37.5\%, 43.75\%\}$; and image C has the same color histogram as image B. If we use the Euclidean distance metric to calculate the histogram distance, the distance between images A and B for GCH is:

$$d_{GCH}(A, B) = \sqrt{(0.25 - 0.1875)^2 + (0.25 - 0.375)^2 + (0.5 - 0.4375)^2} = 0.153;$$

the distance between images A and C equals the distance between images A and B; and the distance between images B and C is zero.

The GCH is the traditional method for color-based image retrieval. However, it does not include information concerning the color distribution of the regions, so the distance between images sometimes cannot show the real difference between images. For example, the distance between images A and C should be smaller than the distance between images A and B, but using the GCH we obtain the same distance. Moreover, in the case of a GCH, it is possible for two different images to have a very short distance between their color histograms (such as in the above example in which image B and image C are treated as the same). This is the main disadvantage of GCHs.

2.4.2 The Local Color Histogram

This approach (referred to as LCH) includes information concerning the color distribution of regions. The first step is to segment the image into blocks and then to obtain a color histogram for each block. An image will then be represented by these histograms. When comparing two images, we calculate the distance, using their histograms, between a region in one image and a region in same location in the other image. The distance between the two images will be determined by the sum of all these distances. If we use the square root of Euclidean distance as the distance between color histograms, the distance metric between two images Q and I used in the LCH will be defined as:

$$d_{LCH}(Q, I) = \sum_{k=1}^{M} \sqrt{\sum_{i=1}^{N} (H_Q^k[i] - H_I^k[i])^2}$$

where M is the number of segmented regions in the images, N is the number of bins in the color histograms, and $H_Q^k[i]$ ($H_I^k[i]$) is the value of bin i in color histogram H_Q^k (H_I^k), which represents the region k in the image Q (I).

The following examples use the same images—A, B and C in Figure 2.4—to show how a LCH works and illustrate how we segment each image into 4 equally sized blocks.

For the LCH, the distance between image A and B (see Figure 2.5) is calculated as follows:

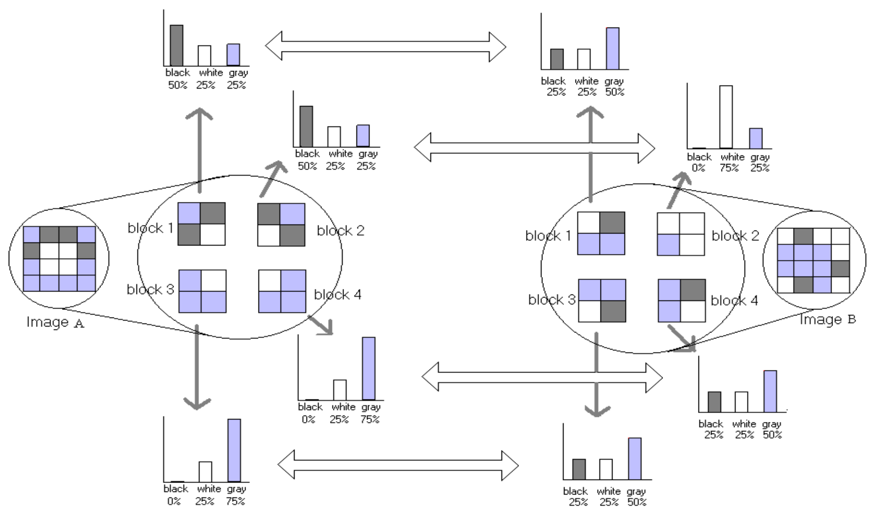


Figure 2.5: Using LCH to compute the distance between images A and B, $d_{LCH}(A,B) = 1.768$, $d_{GCH}(A,B) = 0.153$

$$\begin{aligned} d_{LCH}(A,B) = \; & \sqrt{(0.5-0.25)^2+(0.25-0.25)^2+(0.25-0.5)^2} \\ & \sqrt{(0.5-0)^2+(0.25-0.75)^2+(0.25-0.25)^2} \\ & \sqrt{(0-0.25)^2+(0.25-0.25)^2+(0.75-0.5)^2} \\ & \sqrt{(0-0.25)^2+(0.25-0.25)^2+(0.75-0.5)^2} \quad = \quad 1.768 \end{aligned}$$

Figure 2.6 shows the use of the LCH to compute the distance between images A and C (0.707), while Figure 2.7 shows its use to compute the distance between images B and C (1.768).

In some scenarios, using LCHs can obtain better retrieval effectiveness than using GCHs. The above examples show that the LCH overcomes the main disadvantage of the GCH, and the new distances between images may be more reasonable than those obtained using the GCH. However, since the LCH only compares regions in the same location, when the image is translated or rotated, it does not work well.

For example, in Figure 2.8, if image D is rotated by 90^o, we get image D'. We can then see that image D' is very similar to image E, with only two blocks being different. The distance between images D and E should equal the distance between images D' and E. However, using LCH, the distance between images D and E will be greater than the distance between images D' and E, and this hardly reasonable. The reason for this discrrepancy is that the LCH compares blocks only in the same location, but not necessarily in the proper locations. For example, using LCH, the north-west block in image D is compared with the north-west block in image E but, in fact, the north-west block in image D should be compared with the north-east block in image E. The approach we use in order to overcome this shortcoming of LCHs, will be discussed in detail in the next chapter.

LCHs incorporate spatial information by dividing the image with fixed block segmentation. Some other approaches combine spatial information with color, using different partition schemes [SC96a] [SD96] [HCP95].

2.5 Related work

Smith and Chang [SC96a] developed a color-set back-projection technique to extract color regions, and implemented this technique in the VisualSEEK system. Color sets provide a compact alternative to color histograms by binary vectors, and ignore colors whose value in the corresponding color histogram is below a given threshold (emphasizing only on main colors). The distance of color sets was calculated using a Quadratic-form distance metric.

Smith and Chang transformed the RGB color space into HSV color space and quantized the HSV color space

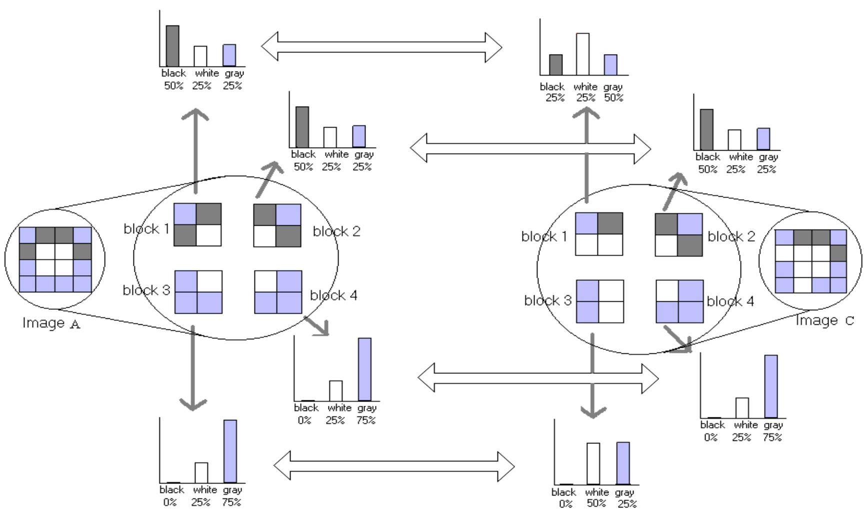


Figure 2.6: Using LCH to compute the distance between images A and C, $d_{LCH}(A, C) = 0.707$, $d_{GCH}(A, C) = 0.153$

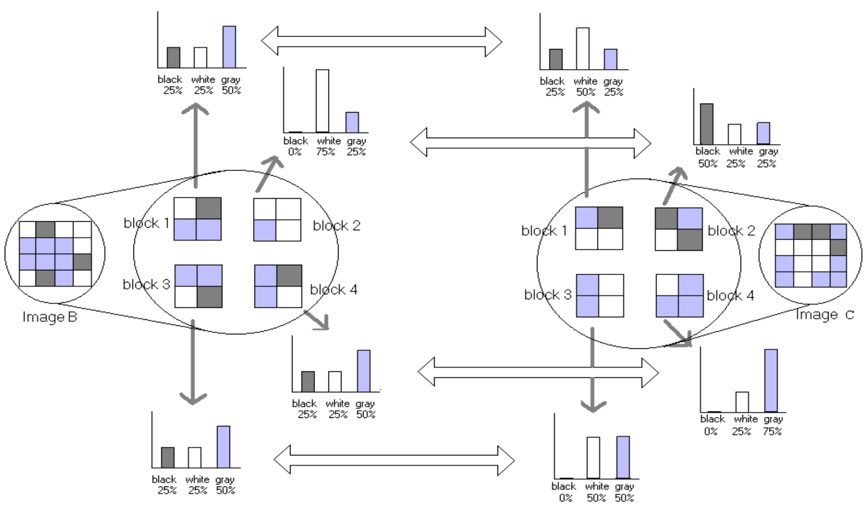


Figure 2.7: Using LCH to compute the distance between images B and C, $d_{LCH}(B, C) = 1.768$, $d_{GCH}(B, C) = 0$

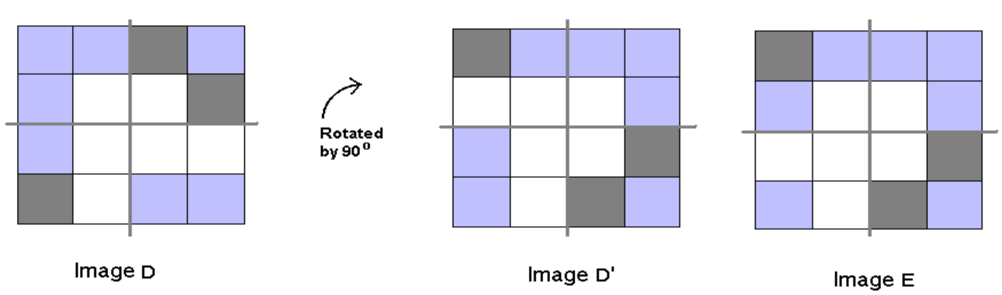


Figure 2.8: An example showing that the LCH fails to compare images D and E

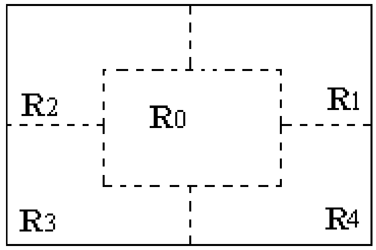


Figure 2.9: 5 regions

to a set of colors. A color median filter was then used to blur images in order to reduce noise. In the next stages, a bi-level image for each color was generated, and a back-projection technique was applied for exacting regions by analyzing the bi-level images through a set of thresholds, i.e., the minimum region size, the required number of pixels in each region for each selected color, the minimum proportion in each region for each selected color, the minimum number of pixels in whole images for each selected color, and the required number of pixels without allocating to any regions for each selected color.

After the segmentation of images, a color set, region area, and location information were stored as features of regions. The distance between two images is the sum of the color set distances and the spatial distances between their regions. Since the color set were consisted of binary vectors, a binary search tree was constructed as the indexing structure.

Stricker and Dimai [SD96] partitioned an image into five partially overlapped, fuzzy regions (see Figure 2.9), and for each region they computed its three Color Moments. The Color Moments for each channel are computed using HSV color space. They slightly modified the formulae to calculate the three moments (Section 2.3.3) by including a weight that is the returned value of the membership function for each pixel, and thus, the pixels close to the border of a region will then have a smaller weight than the pixels in the center of a region. This approach can handle rotations of 90^o, 180^o, and 270^o. As well, because regions are overlapped and fuzzy, this technique is less sensitive to rotations and translations.

Hsu et al [HCP95] captured the spatial information of different colors in images, and retrieved images based on the integrated Color-Spatial information. First, they selected a set of representative colors from the image, according to two presumptions: a color that occupies a large area in the image is an important color; and the center of the image is more important than its surrounding parts. Two color histograms were used for selecting the representative colors: the color histogram for the entire image, and the color histogram for the predefined central part. An heuristic was then used to select colors based on these two color histograms.

As a second step, Hsu et al used the maximum entropy algorithm to partition images into rectangular regions. The algorithm has two parameters: $SIZE$ specifies the minimum area of a region, and $TOLERANCE$ specifies the minimum percentage of a selected dominant color in each region. After the second step, an image is represented as a $\{(c_1, r_1)$, ..., (c_i, r_i), ..., $(c_N, r_N)\}$ where c_i denotes a select color, r_i denotes a list of regions of color c_i, and N denotes the number of colors. Each r_i is a list of tuples that denotes a rectangular region with (x_i^{min}, y_i^{min}) as the lower-left corner and (x_i^{max}, y_i^{max}) as the upper-right corner. The similarity between two images is the degree of overlap between regions of the same color. These experiments on a database with 260 images give better results than are obtained with color histograms.

There are other approaches for incorporate spatial information with color, other than partitioning an image into regions, Pass et al [PZM96] proposed a histogram-refinement approach using so-called Color Coherent Vector (referred to as CCV). In CCVs, when pixels are assigned to bins in histograms, they are split into classes based upon spatial coherence. Two images with identical global color histogram can have different CCVs, and thus the retrieval results can be improved.

This technique classifies pixels as either coherent or incoherent. A coherent pixel belongs to a connected region, in which all the pixels are in the same color, while an incoherent pixel is not. The first step in using CCV's is to blur images, the second is to quantify the color space, and the third is to classify the pixels as either coherent or incoherent. The contiguous regions are computed by connected components. A connected component is a maximal set of linked pixels that are of the same color. When the size of a contiguous region is larger than a fixed value T (T was set as 1% of the image's size), all pixels in this region are coherent, otherwise not.

After the histogram refinement, for a given discretized color, the pixels with this color will be either coherent or incoherent. The number of coherent pixels of the discretized color i is denoted as C_i, while the number of incoherent pixels of the discretized color i is denoted as U_i . (C_i, U_i) is termed the coherent pair for color i, and the color coherent vector is a vector of coherent pairs. The CCV represents an image with $\{(C_1, U_1)$, (C_2, U_2), ..., (C_i, U_i), ..., $(C_N, U_N)\}$.

Using CCVs, the distance between two images is defined as:

$$d_{CCV}(Q, I) = \sum_{i=1}^{N}(|C_i^Q - C_i^I| + |U_i^Q - U_i^I|)$$

where Q and I are two images and N is the number of bins in the histograms.

CCVs use spatial information to refine color histograms, and experiments using them have shown much better performance than color histograms.

Huang et al [HKM$^+$97] proposed a statistical method. They defined a new color feature called the color correlogram that is a table containing color pairs, in which the k^{th} entry for $< i, j >$ specifies the probability of locating a pixel of color j at the distance k from a pixel of color i in the image. This approach is insensitive to the large changes in appearance and shape caused by the changes of viewing positions, camera zooms, etc.

First, a color co-occurrence matrix was constructed. Then the auto-correlogram (the auto-correlogram only captures spatial correlation between same colors) and correlogram were used as the similarity measures. In experiments, they quantized the RGB color space into 64 colors with smoothing images at the beginning. Only the autocorrelogram was computed with the distance set of D={1,3,5,7} in images. The experimental results showed that this approach was more robust than conventional approaches (GCH and CCV).

Chitkara [Chi01] proposed a technique to represent color features of images based on signature bit-strings and GCHs. First, they quantize images into a fixed number of colors and get GCHs of images. Each color element in GCHs is then discretized into binary bins. They emphasized on less dominant colors by setting the binary bins with different capacity. The distance of images is computed using Minkoski-form distance metrics based on difference of signatures. This technique outperforms GCHs in terms of retrieval effectiveness and still saving quite a few storage spaces.

Stehling et al [SNF00] proposed a technique named based on color histograms. Instead of using a single color histogram or fixed number of cells, they use a variable number of histograms called color-shape histograms (CSHs), depending only on the actual number of colors, to represent color features of images. A CSH is used to represent the spatial distribution of a given color for each image cell. In this way, a non-exist color of an image will not be presented. The experiments show that this approach generats outperforms GCHs and LCHs in terms of required results with low space overhead.

Chapter 3

The Harbin Approach

3.1 Motivation

The two traditional color-based image retrieval techniques, GCHs and LCHs, are the starting points of our approach.

GCHs obtain only one color histogram for each image, and no regional information is included. As such, the retrieval effectiveness of GCHs is usually limited. For example, in Figure 3.1, image B and image C have the same color histogram; hence, the distance between image B and C using GCHs is 0, even though these two images are visually different.

The LCH approach involves three steps: (1) segmenting images into blocks and obtaining local color histograms for each block; (2) comparing blocks in the same locations of the two images (the distance between two blocks is the distance between their color histograms); and (3) summing the distances of all the blocks. For example, Figure 3.2 demonstrates how the LCH approach works. Using the LCH approach, the distance between image B and image C becomes non-null, which is more reasonable.

However, in some scenarios—such as when rotating or translating an image into a new image—all blocks in the new image will change their locations, and thus it will be unreasonable for LCHs to simply compare blocks at the same locations. For example, in Figure 3.3, the distance between image D and image E is calculated using the LCH, and this distance is 1.768. If we rotate image E by 90^o, (see Figure 3.4), we obtain image D'.

From Figure 3.4, we can see that image D' is very similar to image E, with only two blocks being different. However, the LCH approach does not compare the blocks in the proper locations. This example not only shows

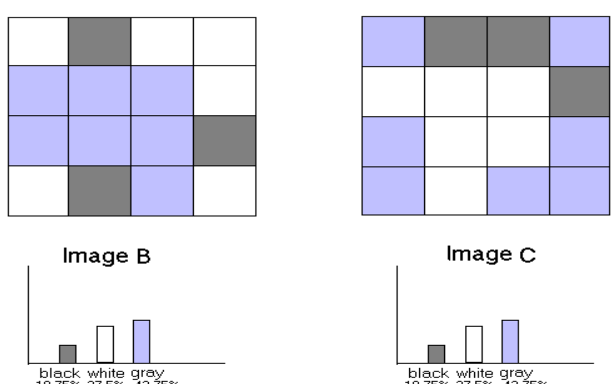


Figure 3.1: Two images and their global color histograms

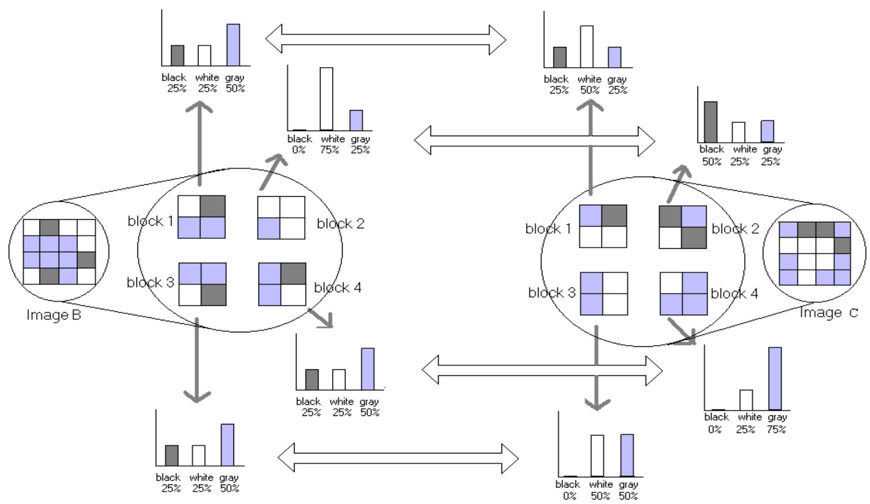


Figure 3.2: How the LCH approach works

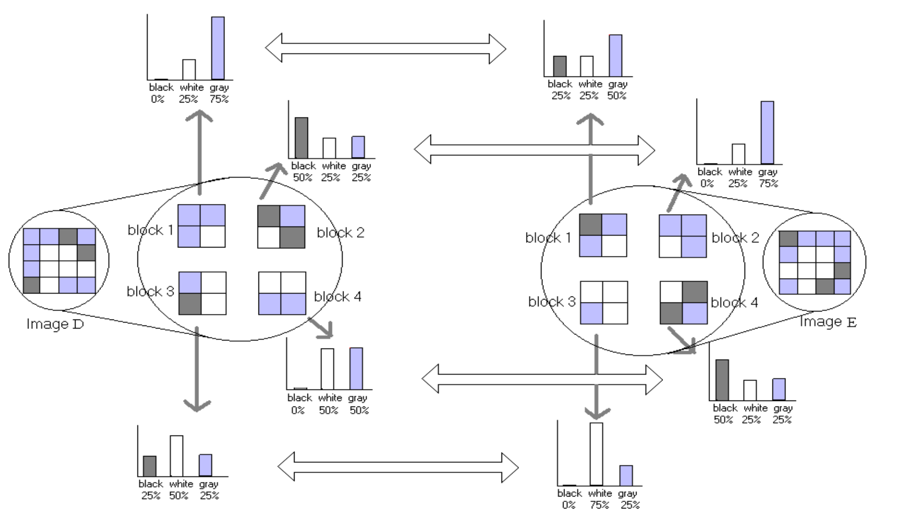


Figure 3.3: An Example to show how the LCH fails

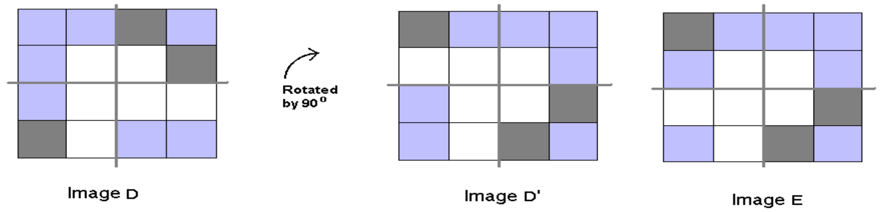


Figure 3.4: Image D is rotated

the limitation of the LCH, but raises the question: "Can we compare blocks in different locations properly?" Our proposed approach aims at answering this question by constructing a weighted bipartite graph and searching for the minimum or maximum cost matching.

3.2 The Harbin Approach for CBIR

Techniques for finding matchings in bipartite graphs have been utilized in many practical applications, such as assignment problems. An assignment problem asks for the best assignment of a set of workers to a set of jobs, with each worker being assigned to one job and each job being finished by one worker until no more workers can be assigned to any jobs. Each worker can finish each job with a "value" (the value could be the productivity or the cost). We wish to make the assignments so that we can maximize the sum of these values (when the value is productivity) or minimize the sum of these values (when the value is cost).

This section shows how to model the problem of calculating the distance between images in terms of finding minimum or maximum cost matchings in bipartite graphs. As far as we know, this is an original contribution in the area of CBIR. We call our new approach, Harbin.

First, we need some definitions [Die97]:

- Bipartite graph - The vertices of a bipartite graph $G(X, Y, E)$ can be partitioned into two sets of nodes, X and Y, so that no edges in set E connect two vertices in the same set of nodes.
- Completed Bipartite graph - A bipartite graph $G(X, Y, E)$ in which the number of edges equals to $|X| \times |Y|$.
- Matching - A matching M of a bipartite graph G is a subset of edges with the property that no two edges in M have a common vertex.
- Maximal matching - A matching is maximal if it is not properly contained in any other matching.
- Perfect matching - If every vertex of a bipartite graph G is incident to some edge of the matching M, then M is called a perfect matching.
- Weighted bipartite graph - A bipartite graph that has a nonnegative weight associated with each edge.
- The cost of a matching - For a weighted bipartite graph, the cost of a matching is the sum of the weights of the edges contained in the matching.
- Minimum cost matching - For a weighted bipartite graph, among all maximal matchings, this is the one that has the minimum cost.
- Maximum cost matching - For a weighted bipartite graph, among all maximal matchings, this is the one that has the maximum cost.
- Minimum cost perfect matching - For a weighted graph, among all perfect matchings, this is the one that has the minimum cost.

Figure 3.5 shows examples of some of these definitions.

The two problems of finding the minimum cost matching and finding the maximum cost matching are equivalent. They can be transformed into each other by simply replacing the weight of each edge w_{ij} with $w_{max} - w_{ij}$, where w_{ij} is the weight of the edge connecting the vertex i and vertex j, and w_{max} is the maximum weight of all edges.

Our approach involves three main steps to compare the distance between two images: (1) partitioning images into blocks and obtaining a color histogram for each block, (2) constructing a bipartite weighted graph, (3)

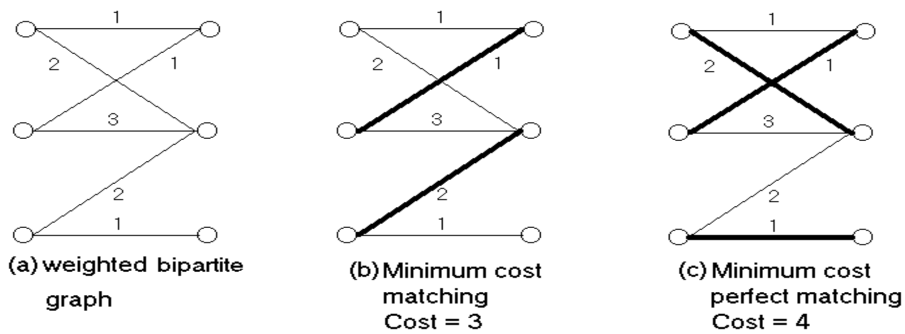


Figure 3.5: Examples of some of the given above definitions

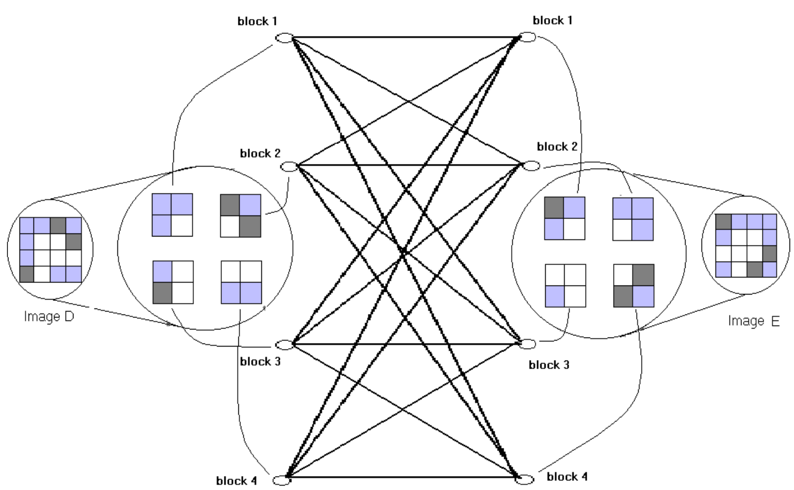


Figure 3.6: How to construct a bipartite graph

searching for the minimum or maximum cost matching. The distance between two images is the cost of the minimum cost matching.

Figure 3.6 shows how we construct a bipartite graph. The first step is the same as the first step for the LCH approach. In the construction of a bipartite graph $G(X, Y, E)$, each block corresponds to a vertex. In this example, we partition each image into four blocks so that the bipartite graph has a total of eight vertices. Each block in one image is connected to all of the blocks in the other image. If the weight of an edge indicates the distance between two blocks connected by the edge, the Harbin approach will search for the minimum cost matching and treat the cost of the matching as the distance between two images. In fact, the similarity between blocks can be calculated using the formula: $s_{ij} = d_{max} - d_{ij}$, where d_{max} is the maximum distance between blocks; blocks i and j are in two images, respectively; s_{ij} is the similarity between the blocks i and j; and d_{ij} is the distance between blocks i and j. If the weight of an edge indicates a similarity between two blocks, the Harbin approach will search for the maximum cost matching and treat the cost of the matching as the similarity between two images.

Figure 3.7 shows a bipartite graph with weights that indicate the distances between vertices. Here, the distance is calculated between the histograms using the Euclidean distance function. The values range between 0 and 1.

After searching for the minimum cost matching in the above bipartite graph, we obtain the minimum cost matching, as shown in Figure 3.8. The dark line indicates an edge contained in the matching. The cost of this

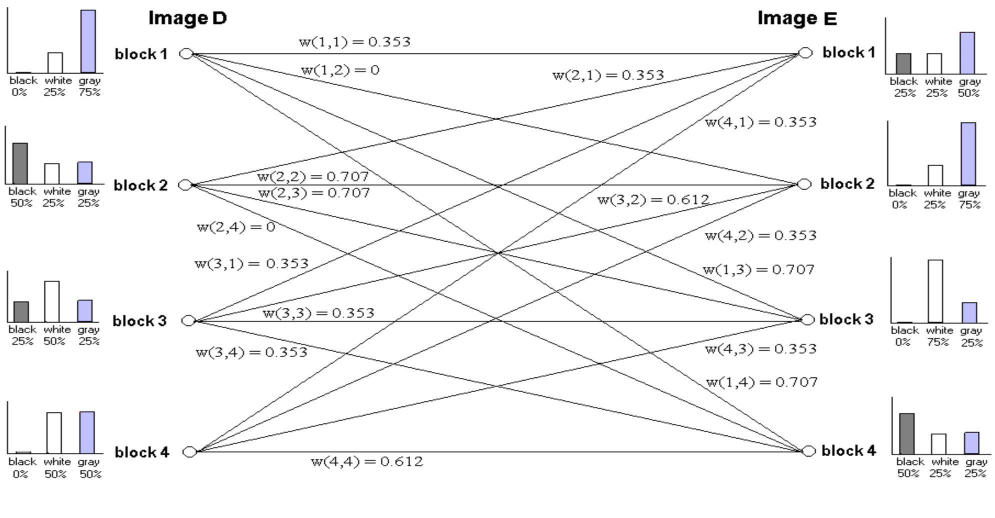


Figure 3.7: The weighted bipartite graph

matching is the sum of the weight of the edges contained in the matching: $Cost = w(1,2) + w(2,4) + w(3,1) + w(4,3) = 0.707$.

Our approach treats this cost as the distance between images D and E.

We do not claim that, since the distance using Harbin is lower than the distance using LCHs, Harbin is more accurate than LCHs; but in this example, Harbin compares the proper blocks between two images, and this is what we seek. For example, block 1 in image D is compared with block 4 in image E, which is what we want.

The Hungarian Algorithm [Kuh55]

The Hungarian algorithm [Kuh55] is a traditional one to find the minimum or maximum cost matching in weighted bipartite graphs. We make use of the Hungarian algorithm in this thesis.

A known bipartite graph $G(X, Y, E)$, $|X| = |Y|$ can be represented by an adjacency matrix $M = (M_{ij})_{n\times n}$ where M_{ij} denotes the weight of the edge between node i in X and node j in Y.

Definition:

- A covering set - A set of rows and columns covering all zeros in a matrix.
- The size of a covering set - The number of rows and columns contained in the covering.

The Hungarian Algorithm for finding the minimum cost matchings in bipartite graphs involves four steps:

1. For each row i, select the minimum value as its boundary-value u_i, and for each column j, set zero as its boundary-value v_j. Construct an excess matrix $B = (B_{ij})_{n\times n}$ where $B_{ij} = M_{ij} - (u_i + v_j)$;

2. Search for the minimum size covering set C of matrix B, and —C— = r; if $r = n$, go on to step 4. Among the uncovered elements, choose the minimum value, which is w_{min}; if both row i and column j have been covered, then set $B_{ij} = B_{ij} + w_{min}$; If both row i and column j have not been covered, then set $B_{ij} = B_{ij} - w_{min}$;

3. Reset the boundary-value; if row i has not been covered, set $u_i = u_i + w_{min}$; if column j has been covered, set $v_i = v_i - w_{min}$; Discard the cover set, and go back to step 2;

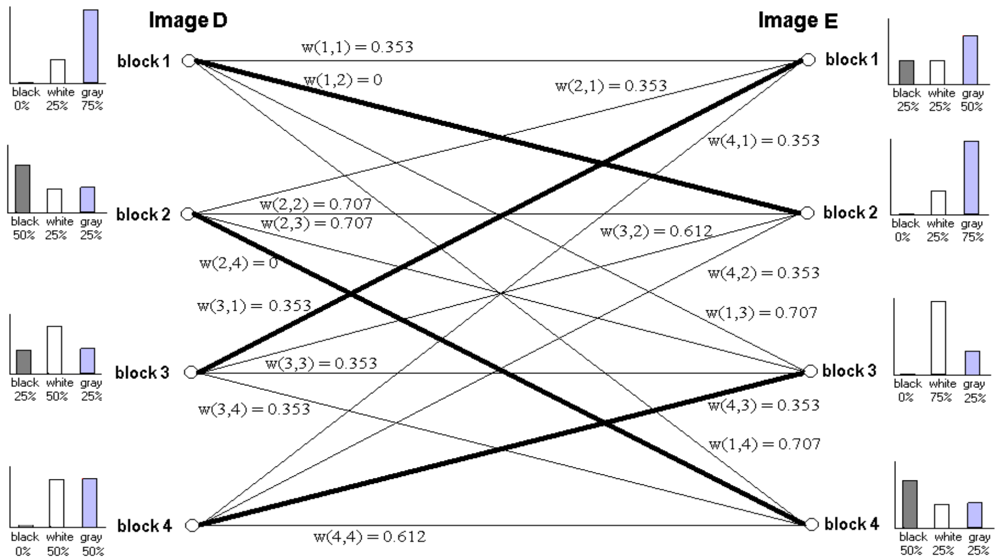


Figure 3.8: A minimum cost matching in the bipartite graph

4. $\sum u_i + \sum v_j$ is the minimum cost, and a transversal of zeros in the covering set corresponds to a minimum cost perfect matching of G.

The Hungarian Algorithm for finding the maximum cost matchings in weighted bipartite graphs involves four similar steps:

1. For each row i, select the maximum value as its boundary-value u_i, and for each column j, set zero as its boundary-value v_j. Construct an excess matrix $B = (B_{ij})_{n\times n}$ where $B_{ij} = (u_i + v_j) - M_{ij}$;

2. Search for the minimum size covering set C of matrix B, and —C— = r; if $r = n$, go on to step 4. Among the uncovered elements, choose the minimum value, which is w_{min}; if both row i and column j have been covered, then set $B_{ij} = B_{ij} + w_{min}$; If both row i and column j have not been covered, then set $B_{ij} = B_{ij} - w_{min}$;

3. Reset the boundary-value; if row i has not been covered, set $u_i = u_i - w_{min}$; if column j has been covered, set $v_i = v_i + w_{min}$; Discard the cover set, and go back to step 2;

4. $\sum u_i + \sum v_j$ is the minimum cost, and a transversal of zeros in the covering set corresponds to a minimum cost perfect matching of G.

The proof of the Hungarian algorithm can be seen in [Kuh55].

Figure 3.9 shows an example of using Hungarian Algorithm to find a minimum cost matching. M is an adjacent matrix corresponding to a bipartite graph. Following the first step of Hungarian Algorithm, we get the excess matrix in Figure 3.9.(1). Rows and columns with the letter C consist of the minimum covering set. The first and fifth columns consist of the minimum covering set of the excess matrix in Figure 3.9.(1), so $r = 2$, where $r = |C|$. Since $(r = 2) < (n = 5)$, where n is the number of rows of the adjacent matrix, we need to repeat the second and third steps of Hungarian Algorithm to modify the excess matrix until $r = n$.

The excess matrix in Figure 3.9.(4) shows $r = n = 5$. One minimum cost matching is $\{M_{13}, M_{25}, M_{34}, M_{42}, M_{51}\}$, and $\sum u_i + v_i = 21$ is the minimum of the corresponding bipartite graph. In fact, a bipartite graph

$$M = \begin{pmatrix} 7 & 6 & 4 & 6 & 1 \\ 4 & 6 & 5 & 7 & 2 \\ 3 & 5 & 7 & 6 & 8 \\ 4 & 7 & 8 & 8 & 5 \\ 2 & 6 & 5 & 6 & 3 \end{pmatrix}$$

$$B = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 2 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} C & & & & C \end{array} \\ \begin{pmatrix} 6 & 5 & 3 & 5 & 0 \\ 2 & 4 & 3 & 5 & 0 \\ 0 & 2 & 4 & 3 & 5 \\ 0 & 3 & 4 & 4 & 1 \\ 0 & 4 & 3 & 4 & 1 \end{pmatrix} \\ \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \end{array} \end{array}$$

(1)

$$B = \begin{array}{c} \\ 3 \\ 4 \\ 5 \\ 6 \\ 4 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} C & & & & C \end{array} \\ \begin{pmatrix} 6 & 3 & 1 & 3 & 0 \\ 2 & 2 & 1 & 3 & 0 \\ 0 & 0 & 2 & 1 & 5 \\ 0 & 1 & 2 & 2 & 1 \\ 0 & 2 & 1 & 2 & 1 \end{pmatrix} \\ \begin{array}{ccccc} -2 & 0 & 0 & 0 & -2 \end{array} \end{array} \begin{array}{c} \\ \\ \\ C \\ \\ \\ \\ \end{array}$$

(2)

$$B = \begin{array}{c} \\ 4 \\ 5 \\ 5 \\ 7 \\ 5 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} C & C & C & & C \end{array} \\ \begin{pmatrix} 6 & 2 & 0 & 2 & 0 \\ 2 & 1 & 0 & 2 & 0 \\ 1 & 0 & 2 & 1 & 6 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \\ \begin{array}{ccccc} -3 & 0 & 0 & 0 & -3 \end{array} \end{array}$$

(3)

$$B = \begin{array}{c} \\ 5 \\ 6 \\ 6 \\ 8 \\ 6 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} & & C & & C \end{array} \\ \begin{pmatrix} 6 & 2 & \underline{0} & 1 & 0 \\ 2 & 1 & 0 & 1 & \underline{0} \\ 1 & 0 & 2 & \underline{0} & 6 \\ 0 & \underline{0} & 1 & 0 & 1 \\ \underline{0} & 1 & 0 & 0 & 1 \end{pmatrix} \\ \begin{array}{ccccc} -4 & -1 & -1 & 0 & -4 \end{array} \end{array} \begin{array}{c} \\ \\ \\ C \\ C \\ C \\ \\ \end{array}$$

(4)

Figure 3.9: Example of Hungrarian Algorithm

can have several minimum cost matchings, but the minimum cost is unique. For instance, $\{M_{15}, M_{23}, M_{34}, M_{42}, M_{51}\}$ is another minimum cost matching, but its cost is still 21.

The time complexity of the Hungarian algorithm is $O(mn^2)$, where m and n are the number of edges and the number of vertices in the original bipartite graph, respectively. Here, since we construct complete bipartite graphs, m equals n^2; hence, the time complexity is $O(n^4)$.

The Hungarian algorithm is the original one for calculating the minimum and maximum cost perfect matching on a weighted bipartite graph. Some other algorithms that formulate matching problems as network flow problems are more efficient than the Hungarian one, including: Edmonds and Karp's Algorithm [EK72], which has the time complexity $O(mn(logn))$; Fredman and Tarjan's algorithm [FT87] (which introduced the Fibonacci heap to Edmonds and Karp's algorithm), which has the time complexity $O(mn + n^2(logn))$; and Gabow's algorithm [Gab82], which has the time complexity $O(mn^{\frac{3}{4}}(logU))$, where U is the greatest absolute value among edge costs.

In this thesis, we implement the Hungarian Algorithm to compute the minimum and maximum cost matching in bipartite graphs.

3.3 Enhancement of the Harbin approach

If two images are similar, they must contain some similar regions (they could be in different locations). If a region in one image is similar to a region in another image; we say that the two regions are matched. We assume that a region in one image can be matched to only one region in another image, and matched regions between two images will determine their distance or similarity, which is computed by summing the distances or similarities between all matched regions. The basic idea behind the Harbin approach is to try to match regions (here regions are blocks) between two images so that we can minimize the distance or maximize the similarity between two images.

When the Harbin approach uses minimum cost matching to calculate the distance between images, the match-

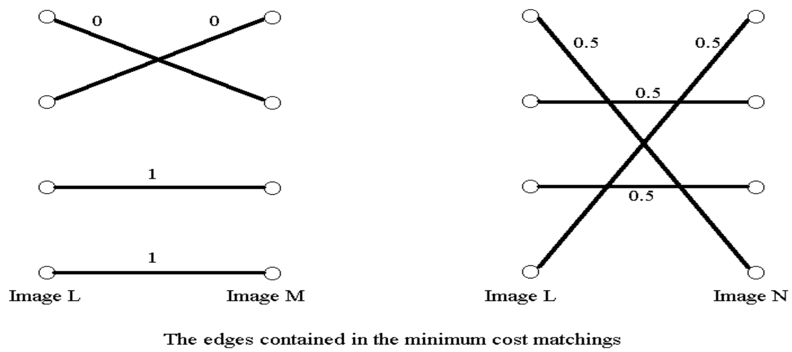


Figure 3.10: Using the minimum cost matching to calculate distances between images L and M, L and N.

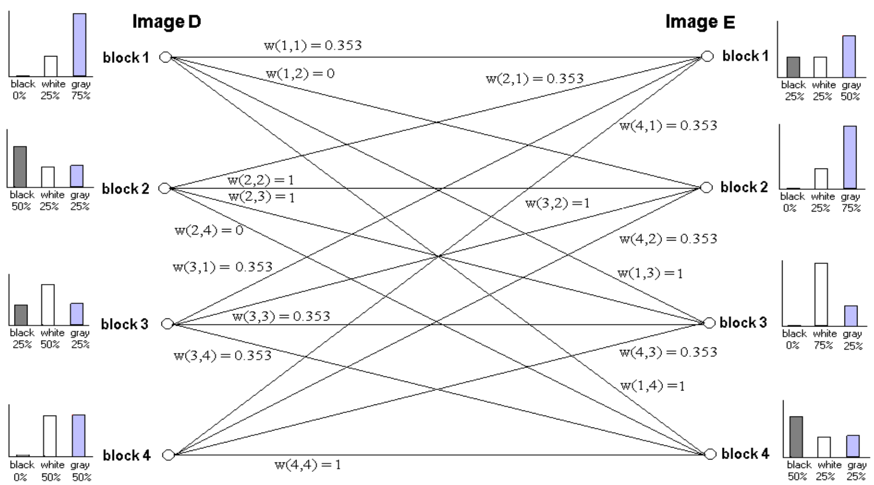


Figure 3.11: The weighted bipartite graph after using Δ = 0.5

ings may contain edges with large distances or small similarities. However, when the distance between two blocks is too large (say, larger than 0.5), the corresponding blocks will be not similar at all, and containing the edges with large distances (or small similarities) will only add noise to the final distance (or similarity) between images. This means that we need not differentiate the ones which are very dissimilar, so it is unnecessary to discriminate between the values of large distances or small similarities. For example, for three given images—L, M, and N—we want to use the Harbin approach to calculate the distance between images L and M and between images L and N, so we construct two bipartite graphs and search for the minimum cost matchings in them. Figure 3.10 shows the edges contained in the minimum cost matchings.

From Figure 3.10, we can see that the distance between images L and M is the same as the distance between images L and N, and the value is 2. But this seems unreasonable (Note that half of image L and half of image N can be exactly the same, but all the 4-pair blocks between image L and N are quite different). The reason for this is that the edges with distance 0.5 add noise to the final answer. In order to reduce the noise, when constructing the bipartite graph, we use an heuristic threshold as a measurement to determine weights of edges, and we call this threshold Δ. If the distance between two blocks is larger than Δ, we simply set the maximum distance as the weight of the corresponding edge in bipartite graphs (here the maximum distance is 1). If we use the distance 0.5 as Δ in bipartite graphs, there will be no edges whose distance is between 0.5 and 1. Figure 3.11) demonstrates the use of 0.5 as Δ on the complete bipartite graph in Figure 3.7.

Figure 3.12 shows the minimum cost matching of the bipartite graph in Figure 3.11, with the dark line

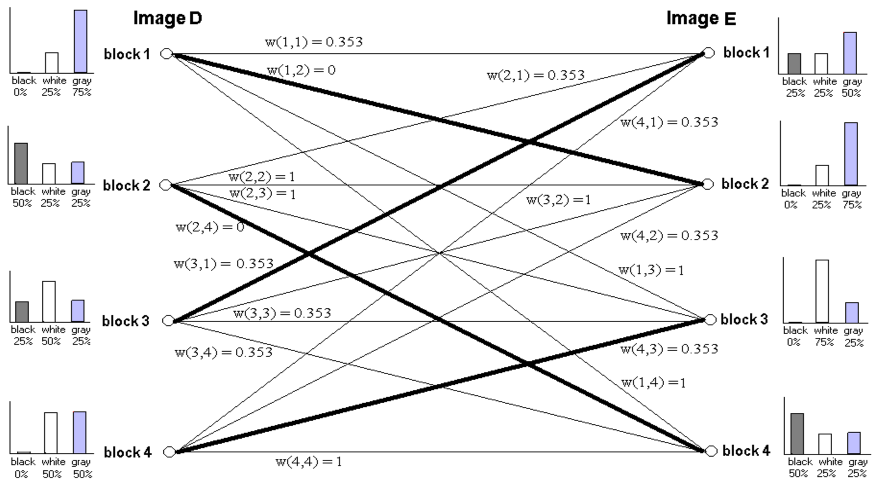


Figure 3.12: The minimum cost matching after using $\Delta = 0.5$

indicating an edge contained in the matching. In this example, since the weights of the edges included in the old minimum cost matching (see Figure 3.8) are less than Δ, after using Δ, the minimum cost matching is the same as for the old one, and the distance between images D and E is still 0.707. However, in general situations, after using Δ, the new minimum cost matchings can become much different.

By setting all the distances larger than Δ as the maximum distance, we hope to reduce the noise and improve retrieval effectiveness. However, if we use too small a value for a Δ, we will discard the edges which connect similar blocks, and this is not what we want, so the problem of how to obtain the value of Δ to get the best retrieval results will depend on the experimental results.

The minimum cost matching problem can be transformed into a maximum cost matching problem by changing the weight of edges from the distance to the similarity, and the Harbin approach can also be used to calculate the similarity between images. The edges contained in the minimum cost matching should be the same as the edges contained in the maximum cost matching.

We can still use Δ to determine the weights of edges to reduce noise. If the distance between two blocks is larger than Δ, we simply set the similarity between two blocks as 0. If an edge has the weight 0, we can discard this edge in bipartite graphs. In this way, we reduce the number of edges in the bipartite graph, and this means that the bipartite graph will not be complete. We then search for the maximum cost matching in incomplete bipartite graphs, and the cost of the matching is treated as the similarity between images. We know that the time complexity of all matching algorithms is related to the number of edges, so after using Δ and searching for maximum cost matching, the Harbin approach becomes more efficient.

For example, after using 0.5 as Δ, and changing the distance to similarity on the complete bipartite graph in Figure 3.8, we obtain the bipartite graph in Figure 3.13.

Figure 3.14 shows the maximum cost matching of the bipartite graph in Figure 3.13, with the dark line indicating an edge contained in the matching. The similarity between images D and E is 3.294. Before searching for minimum cost matching to minimize dissimilarity, we must search for maximum cost matching to maximize similarity.

3.4 Improving Retrieval Efficiency

The Harbin approach overcomes the shortcoming of LCHs, which compare blocks only in the same locations, through its high time complexity algorithm (In this thesis, we use the algorithm that has the time complexity

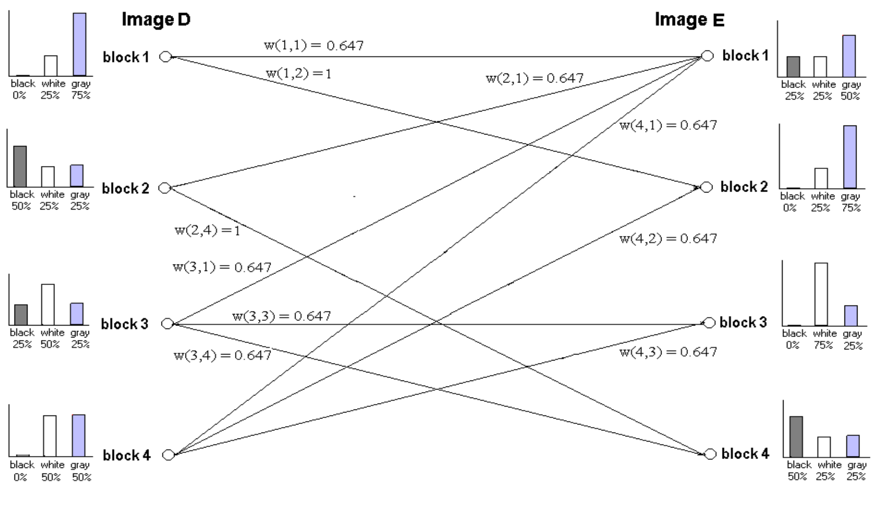


Figure 3.13: Using $\Delta = 0.5$; weights in the graph indicate similarities

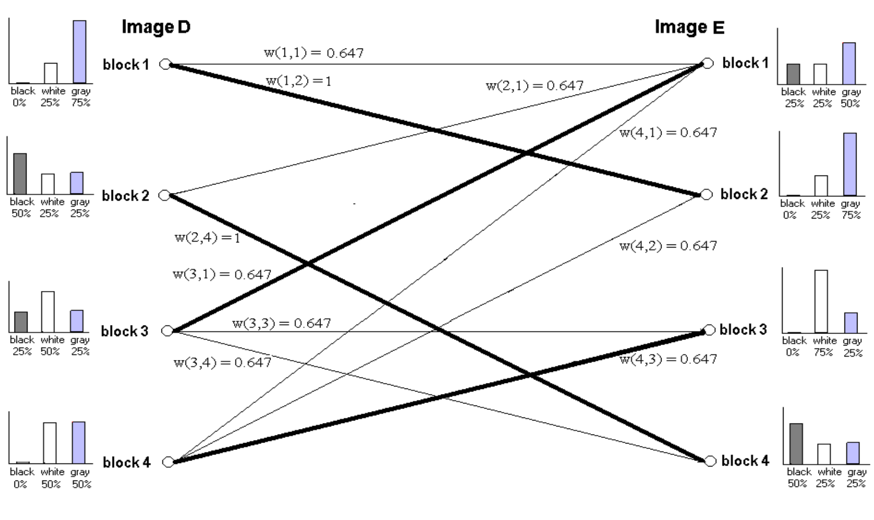


Figure 3.14: A maximum cost matching after using $\Delta = 0.5$

$O(mn^2)$). In comparison with the LCH approach, the Harbin method trades effectiveness for efficiency. In fact, for a huge image database, an efficient indexing structure, which is used to avoid a linear search, is usually more important than the efficiency of the approach itself. Even though the issue of indexing is beyond the scope of this research, the following question is legitimate: without an effective indexing structure, can we still use the Harbin method in practice?

The GCH technique is not very effective because it does not include regional information. If the similarity obtained between two images, using the Harbin approach, is high, the similarity obtained using the GCH technique must also be high. On the other hand, if the similarity obtained using the GCH method between two images is low, the similarity obtained using Harbin must also be low. Hence, it is not necessary to use the Harbin approach for images that have low similarities obtained using the GCH. Based on this idea, we propose to use the Harbin method to refine retrieval results obtained using the GCH. In the beginning, we use the GCH approach to obtain the most desired images (say the top 100 images), and the Harbin technique is then used to re-rank the top images. In this way, instead of using the Harbin method to compare similarities between the query image and all the images in databases, we use it only on the images that have a high similarity obtained using the GCH, and thus we avoid a linear search, which is the main function of an indexing structure. Although the GCH technique is not very effective, and the Harbin method is not very efficient, the new method combines the efficiency of the GCH with the effectiveness of Harbin—so it is hoped that this new method can be both effective and efficient.

Chapter 4

Experimental results

In this Chapter, we evaluate our approach with respect to three parameters: (1) the number of bins in color histograms (i.e., the number of quantized colors), (2) the number of blocks used to segment images, and (3) Δ (i.e., the threshold value to determine weights of edges). We also compare the performances of the two traditional approaches, GCH and LCH, to the new proposed approach, Harbin.

4.1 Experimental Setup

The retrieval performance was evaluated using a test database of 20,009 JPEG images from a CDROM created by Sierra Home. Our query set consisted of 21 query images[1]; each query has a subjective answer set with about 9 images, on average. Fifteen query images and their answer sets were obtained from the CDROM "Corel Gallery Magic 65,000", and the remaining 6 query images and their answer sets were generated by us, using a digital camera. As discussed in Chapter 3, the Harbin approach overcomes a shortcoming of the LCH, which compares blocks only in same locations and thus fails to find rotated or translated images that are similar to query images. However, we assume that answer sets should include those similar images resulting of rotations and/or translations (We generated some answer sets including rotated and translated images). Therefore, the query images can be categorized into two classes: one including 9 query images that were similar to the images in their answer set by considering translation and rotation, and one including 12 query images that did not include rotated and translated images in their answer set.

We implemented all approaches using C++/C, C-Shell script, and ImageMagick API[2]. All the response times of our experiments were measured using a PC running Linux OS with a 550MHz Celeron CPU and 128 Mbytes main memory.

4.2 Experiment evaluation

4.2.1 Precision-Recall graph

Precision-Recall graphs [YN99] (described next) are a standard way to evaluate retrieval results for information retrieval systems, hence, we also use them in our research.

Let R be the set of relevant images in the database, A the answer set of retrieved images, and R_A the set of relevant images in set A (See Figure 4.1)

Recall is the fraction of relevant images in the database that have been retrieved in response to a query, namely, $recall = \frac{|R_A|}{|R|}$. Precision is the fraction of the retrieved images that are relevant to the query image,

[1] All this can be seen at: www.cs.ualberta.ca/~mn/CBIRtwo, and some samples can be also seen in the Appendix.

[2] www.imagemagick.org

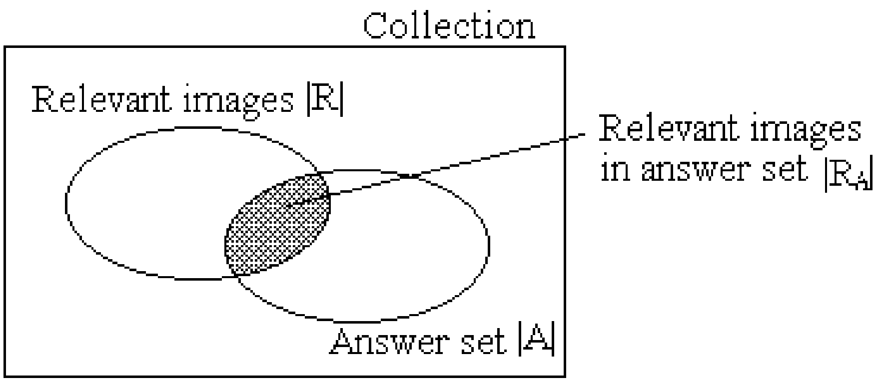


Figure 4.1: Precision and recall for a given query

namely $precision = \frac{|R_A|}{|A|}$. A Precision-Recall curve is usually based on 11 recall levels (0%, 10%, 20%, ..., 100%).

Retrieval effectiveness is always evaluated for more than one query, and as a result, the reported precision for each level is the average precision of all the queries, which is defined as:

$$Precision_{avg}(l) = \sum_{i=1}^{|Q|} \frac{Precision_i(l)}{|Q|}$$

where $Precision_{avg}(l)$ is the average precision at level l, $Precision_i(l)$ is the precision for query i at level l, and $|Q|$ is the number of queries.

4.2.2 Retrieval Effectiveness

Precision-Recall graphs intuitively demonstrate the retrieval quality from two directions, but sometimes it may be difficult to investigate whether one approach outperforms the other. In these situations, a single value for evaluation is required. A strategy presented in [FBF+94] evaluated retrieval quality by comparing the average rank of all relevant images to the ideal average rank (images were ranked according to similarities, and the smaller the rank, the larger the similarity). The word "ideal" means that after retrieval, all the relevant images are ranked on the top. In this thesis, we use the metric based on the average rank and the ideal average rank. We denote the metric for retrieval effectiveness ε, which is defined as:

$$\varepsilon = \frac{\text{The average rank}}{\text{The ideal average rank}}$$

For example, assume we used two approaches, A and B, to retrieve images from a database, and it is known that the image database includes two relevant images for a given query image. If approach A returns the two relevant images ranked 2 and 7, and approach B returns them with ranks of 3 and 9, the average ranks of A and B are $\frac{2+7}{2} = 4.5$, and $\frac{3+9}{2} = 6$, respectively. The ideal average rank should be $\frac{1+2}{2} = 1.5$. The ε values of approach A and B are $\frac{4.5}{1.5} = 3$, $\frac{6}{1.5} = 4$ respectively. For this metric, the smaller the value the greater the effectiveness, and the best retrieval technique will return the $\varepsilon = 1$. Thus, approach A is more effective than approach B. More precisely, for a given query image, the metric can be defined as:

$$\varepsilon = \frac{2 \times \sum_{j=1}^{|R|} Rank_j}{|R| \times (|R|+1)}$$

where $Rank_j$ represents the ranks of relevant images.

Since this measure is normalized in terms of the sizes of the querys' answer sets, we can simply compare the value of this measure among different queries. Therefore, when we use more than one query to evaluate a retrieval approach, the average retrieval effectiveness will be used as the criteria for the overall performance, which is defined as:

$$\varepsilon_{avg} = \sum_{i=1}^{|Q|} \frac{\varepsilon_i}{|Q|}$$

where ε_{avg} is the overall retrieval effectiveness, and ε_i is the retrieval effectiveness for query i.

Techniques	Color 27	Color 64	Color 125
GCH	174.7	124.3	112.7
LCH using 4 by 4 blocks	308	239.7	253.5
Harbin using 4 by 4 blocks, Δ=1.00	144.9	118.7	100.2
Harbin using 4 by 4 blocks, Δ=0.75	148.0	125.2	109.1
Harbin using 4 by 4 blocks, Δ=0.50	150.3	114.6	84.6
Harbin using 4 by 4 blocks, Δ=0.25	136.8	84.5	77.8
LCH using 6 by 6 blocks	360.6	312.3	315.2
Harbin using 6 by 6 blocks, Δ=1.00	144.4	103.5	105.8
Harbin using 6 by 6 blocks, Δ=0.75	146.0	108.0	108.2
Harbin using 6 by 6 blocks, Δ=0.50	139.7	82.0	98.5
Harbin using 6 by 6 blocks, Δ=0.25	114.3	60.5	61.8
LCH using 8 by 8 blocks	337.1	281.0	302.7
Harbin using 8 by 8 blocks, Δ=1.00	118.8	95.7	69.6
Harbin using 8 by 8 blocks, Δ=0.75	119.4	97.8	71.2
Harbin using 8 by 8 blocks, Δ=0.50	103.8	71.1	62.6
Harbin using 8 by 8 blocks, Δ =0.25	79.0	48.4	28.4

Table 4.1: Summary of the experimental results for 21 queries in terms of ε_{avg}

In this thesis, we use both ε and the Precision-Recall graph for evaluating the retrieval effectiveness of the investigated approaches.

4.2.3 Obtained Results

In the experiments, all three approaches (the GCH, LCH, and Harbin) are performed on the RGB color space. We used three choices for quantizing the color space: 27 colors, 64 colors, and 125 colors. Other choices could also be made. However, as discussed in Chapter 2, there is a trade-off in determining how many colors should be used; using more colors may not generate better retrieval results. On the other hand, using more colors will require more storage space for metadata, and the size of the metadata can influence the retrieval efficiency. For instance, after keeping all metadata in the main memory, we could improve retrieval efficiency by saving time on accessing the disk. As well, using more colors will increase the time needed to calculate the distance between color histograms.

For the LCH and Harbin, we used three choices for the image partition scheme: images were evenly divided into 4 by 4 blocks, 6 by 6 blocks, and 8 by 8 blocks. In the Harbin approach, after using more blocks, the constructed bipartite graph will contain more vertices, and more time will be spent to compute the minimum or maximum cost matchings (the efficiency of algorithms to solve the matching problem is always related to the number of vertices, such as in the Hungarian algorithm, which has the time complexity $O(mn^2)$). In addition, when more blocks are used, more storage space for metadata will be required so as to influence the retrieval efficiency. We use four choices for Δ: 1.0, 0.75, 0.5, 0.25 (this applies for the Harbin approach only).

Table 4.1 lists the experimental results of the three approaches in terms of ε_{avg}, for the 21 queries under every combination of the three choices for the number of bins in color histograms, the three choices for numbers of blocks, and the four choices for Δ.

We can observe that, in the Harbin approach, using more colors, more blocks, and a smaller Δ yields better retrieval results. The best results occur when using 125 colors, 8 by 8 blocks, and Δ=0.25 respectively. In order to better understand the relationships between the parameters and the retrieval results, we draw 3 ¡parameter¿-ε_{avg} graphs and 3 Precision-Recall graphs. Each of these graphs reflects the retrieval performance obtained by

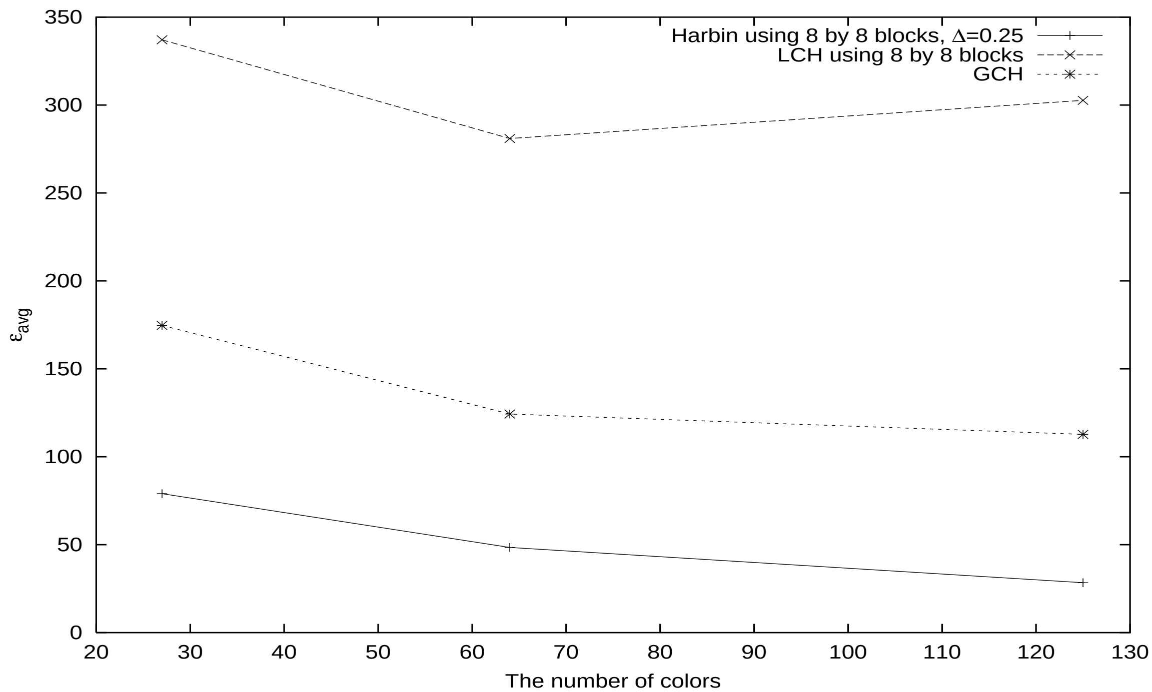


Figure 4.2: The relationship between colors and ε_{avg}

changing one parameter and keeping the others constant.

Figure 4.2 shows the relationship between the number of colors and ε_{avg} for 21 queries when only the number of colors (27, 64, 125) is changed, and 8 by 8 blocks and Δ=0.25 are kept. From this figure, we can observe that when the same parameters are used, the Harbin approach always produces better ε_{avg} than the GCH and LCH approaches. (In fact, all the results of the Harbin approach are much better than the best results of the other two approaches). Figure 4.3 shows Precision-Recall curves in which we can observe that, for the Harbin approach, using 125 colors yields the best Precision-Recall relationship. In addition, in comparision to the curve generated by the GCH using 125 colors, and the curve produced by the LCH using 125 colors and 8 by 8 blocks, the Harbin approach always enhances the Precision-Recall relationship. Therefore, for the Harbin approach only, when the number of blocks and Δ are not changed, we prefer to use 125 colors; when the same parameters are used, the Harbin approach is the best.

Figure 4.4 shows the relationship between the number of blocks and ε_{avg} for 21, queries when the number of blocks are changed (4 by 4 =16 blocks, 6 by 6 = 36 blocks, 8 by 8 = 64 blocks), and 125 colors and Δ=0.25 are used. From this figure, we can observe that the Harbin approach still produces better ε_{avg} than the GCH and LCH methods. With the Harbin approach, the retrieval result is best when 64 blocks are used. Figure 4.5 demonstrates the same observation through Precision-Recall curves. This graph also shows that the Harbin generates better Precision-Recall relationships than the GCH and LCH approaches.

From Table 4.1, we observe that when Δ = 0.25 (the smaller of the values tested) ε_{avg} is more accurate. However, if we use a Δ which is too small, when we construct bipartite graphs, we will discard the edges which connect similar blocks, and this is not what we want. In order to better understand the relationship between Δ and the retrieval results, we performed more experiments using additional for Δ around 0.25. Figure 4.6 shows the relationship between Δ and ε_{avg} for 21 queries when Δ is changed (0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.50, 0.75, 1.00), and 125 colors and 8 by 8 blocks are used. From this figure, we can observe that using a reasonably small Δ will improve ε_{avg}. When using 125 colors and 8 by 8 blocks, the Δ that produces the best retrieval results should be approximately 0.25. Figure 4.7 demonstrates the same concept using Precision-Recall curves.

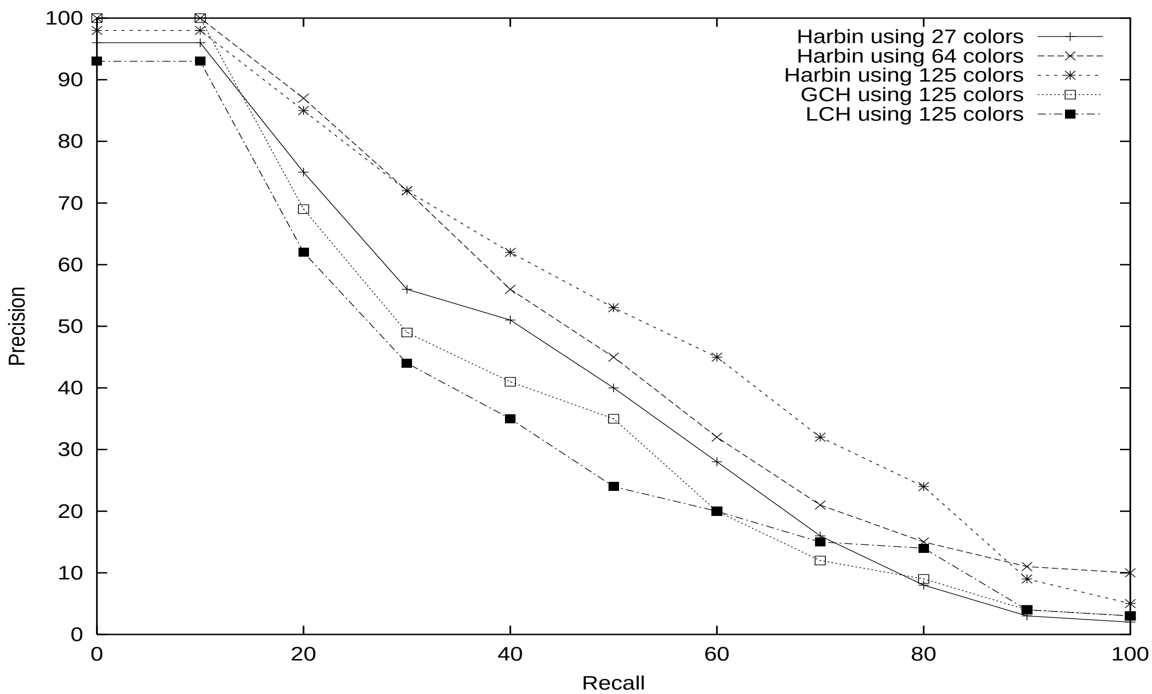


Figure 4.3: A Precision-Recall graph using 8 by 8 blocks and Δ=0.25

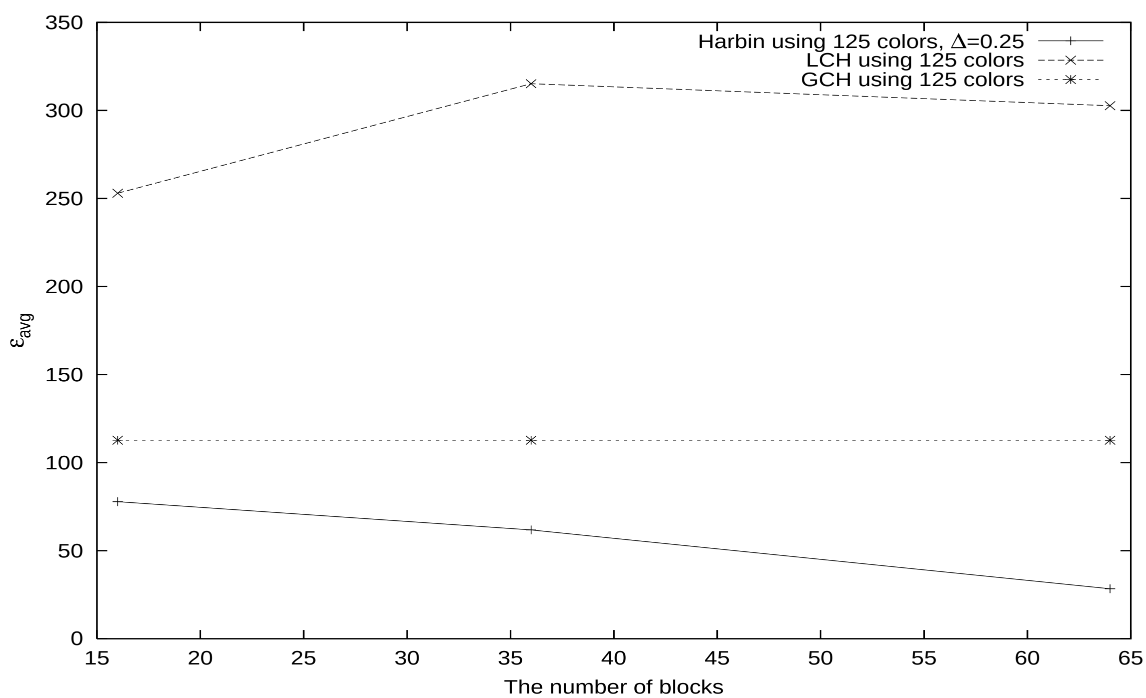


Figure 4.4: The relationship between the number of blocks and ε_{avg}

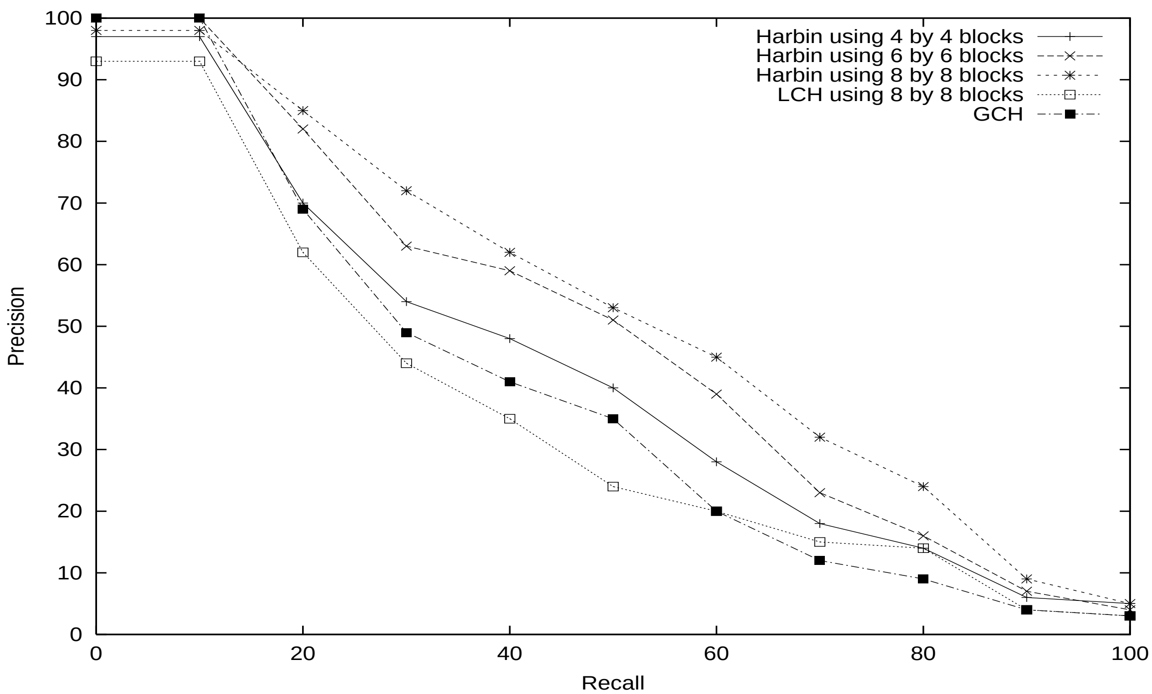


Figure 4.5: A Precision-Recall graph using 125 colors, Δ=0.25

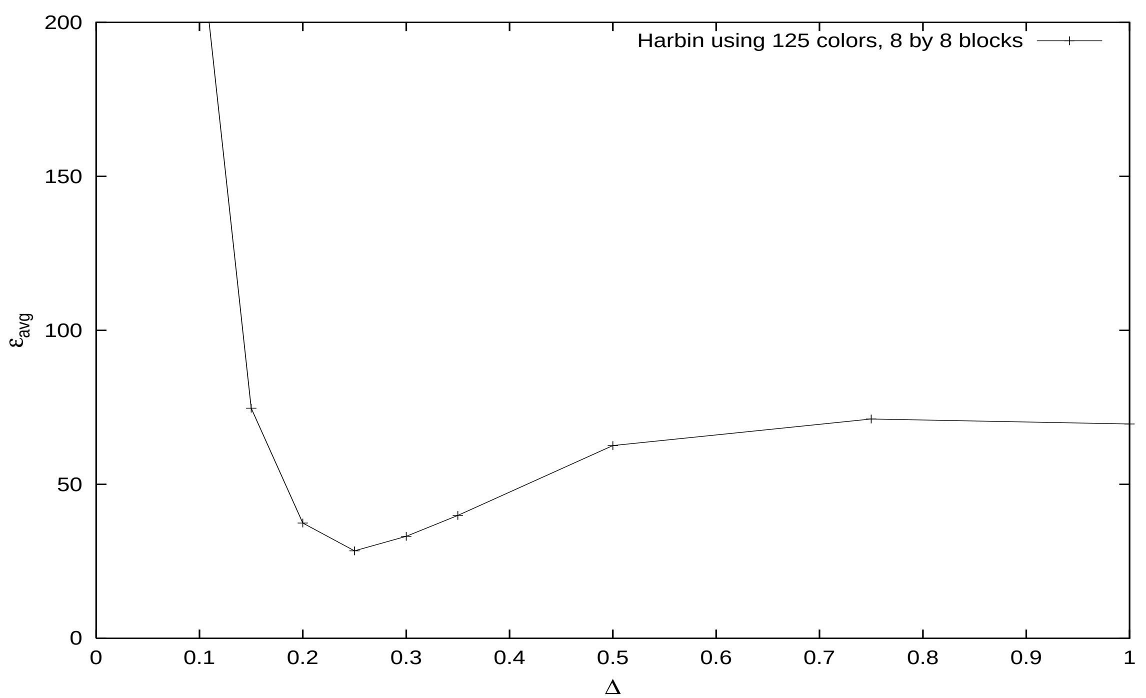


Figure 4.6: The relationship between Δ and ε_{avg}

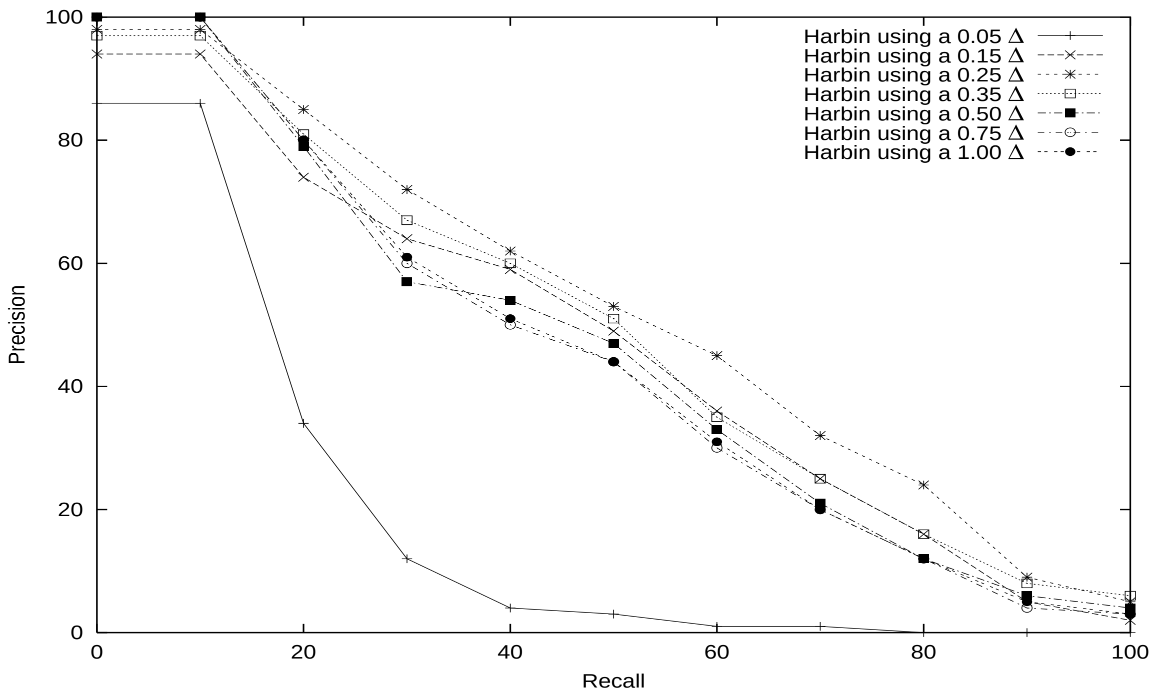


Figure 4.7: A Precision-Recall graph using Harbin with different Δ

Categories	GCH	LCH	Harbin
1 (9 queries)	182.6	748.0	40.3
2 (12 queries)	61.0	16.4	21.6

Table 4.2: Retrieval effectiveness results for the two categories using 125 colors, 8 by 8 blocks, and Δ=0.25 in terms of ε_{avg} (Category 1 is the one with rotated/translated images).

Table 4.1, and Figures 4.2, 4.3, 4.4, 4.5, show that retrieval results using the LCH are even worse than for the GCH, even though the LCH was designed to improve the GCH's retrieval effectiveness. The reason for this was discussed in Section 4.1: there are 9 queries (out of the 21) that expect rotated and translated images in their answer sets, and the LCH approach will fail for those queries. This causes the overall performance of the LCH to deteriorate.

Table table:comparing GLH on two catagories shows the retrieval results of the three approaches for the two categories separately (i.e., sets with and without rotated/translated images) using 125 colors, 8 by 8 blocks, and Δ=0.25. From this table, we observe that for Category 1, the Harbin approach yields significantly better retrieval effectiveness than the GCH and LCH approaches, however, the LCH approach loses effectiveness much more than the GCH approach. This confirms that the LCH is bound to fail for the queries in Category 1. We can claim, therefore, that the Harbin approach is the best choice for queries that expect rotated and translated images in their answer sets. Figure 4.8 compares the three approaches for the queries in Category 1 using the Precision-Recall graph. From the graph, we can also observe that the Harbin approach has the best Precision-Recall relationship. Table 4.3 compares the three approaches for the 9 queries in Category 1. From the table, we can see that the Harbin approach not only improves the average retrieval effectiveness but also is successful in 7 queries out of 9.

Table table:comparing GLH on two catagories also shows the retrieval results for Category 2. The LCH approach obtains the best ε_{avg}, followed by the Harbin approach. In fact, the retrieval effectiveness is not only related to the retrieval techniques but also depends on the queries and their answer sets. Because in Category 2, the query images and the images in their answer sets always have similar layouts (no rotatation and translation),

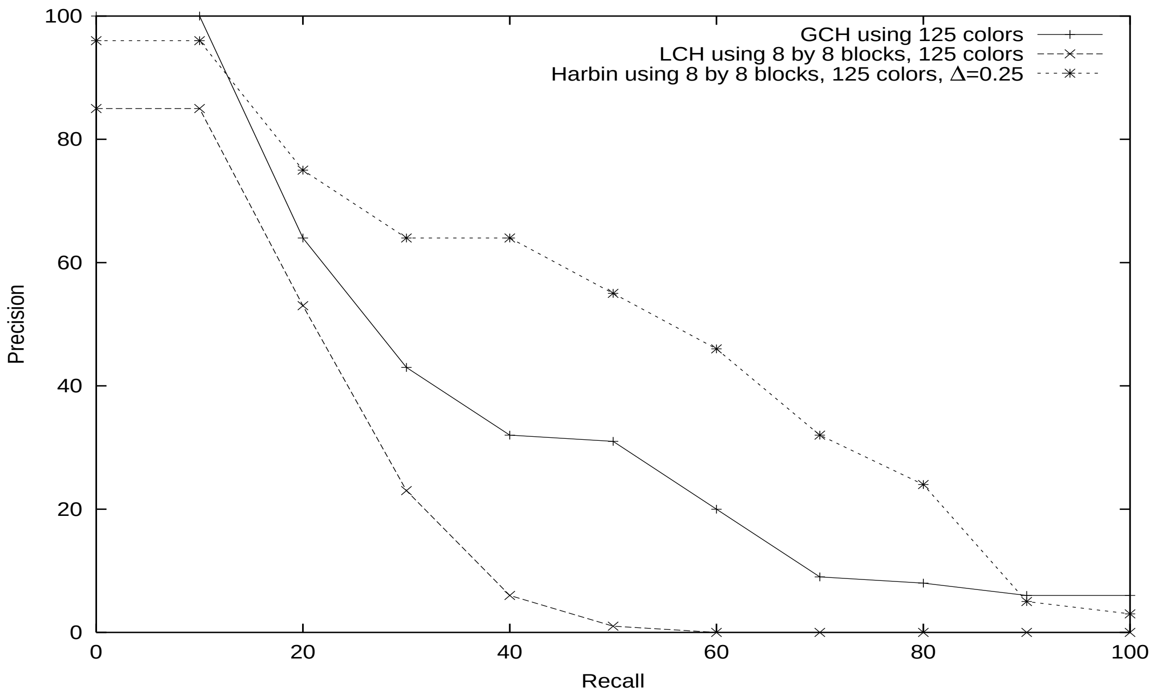


Figure 4.8: A Precision-Recall graph comparing the three approaches for queries whose answer sets have rotated and translated images

Queries	Harbin	LCH	GCH
bonzai	4.5	197.3	48.8
building	15.2	146.0	129.4
bus	85.4	701.5	1037.2
coke	47.5	1181.1	201.4
door	2.3	77.4	1.3
horse	9.0	447.4	62.3
mouse	70.0	514.6	80.2
oldtrain	101.7	2291.6	34.0
painting	27.1	1175.8	49.1
Average	40.3	748.0	182.6
Number of wins	7	0	2

Table 4.3: A comparison of the three approaches for 9 queries in the first category (including rotated/translated images) using 125 colors, 8 by 8 blocks, and Δ=0.25

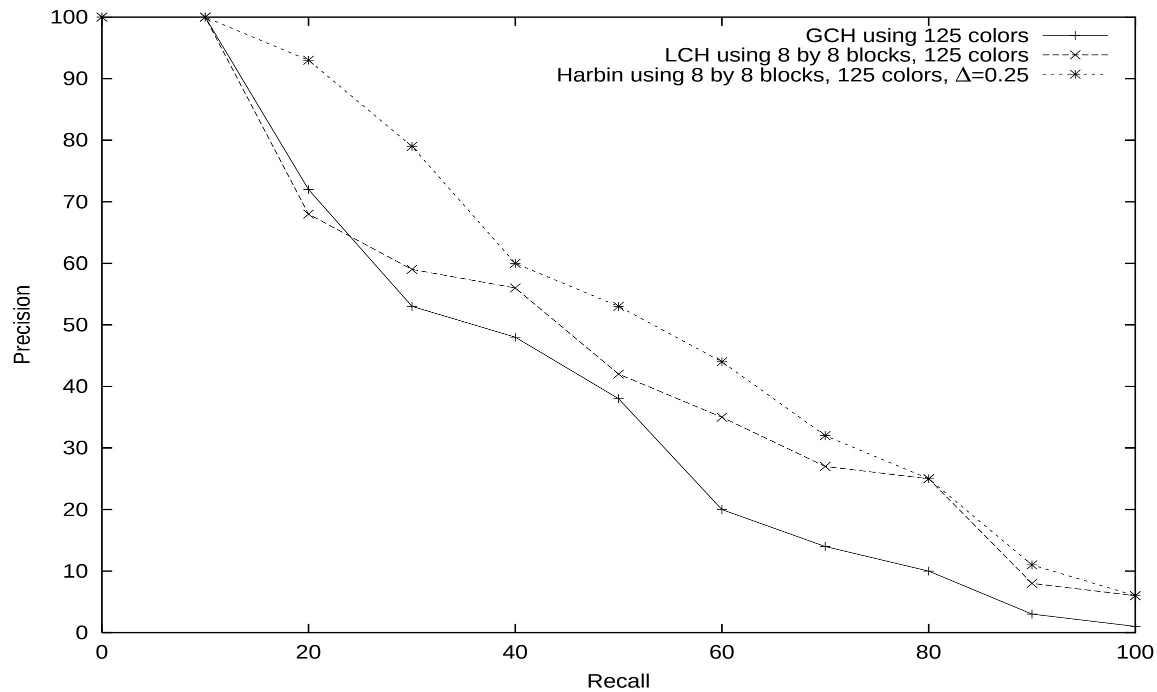


Figure 4.9: A Precision-Recall graph comparing the three approaches for queries whose answer sets do not include rotated and translated images

the LCH approach generates the best results. On the other hand, since the retrieval results of the Harbin approach are unrelated to the image layout, it still yields good results. Figure 4.9 compares the three approaches for the second categories using a Precision-Recall graph. It shows that the Harbin approach has the highest curve, which means that the Harbin method obtains the best Precision-Recall relationship. Table 4.4 compares the three approaches for the 12 queries in Category 2. From the table, we can see that the Harbin approach equals the LCH approach by being successful in 6 queries out of the 12. Therefore (although only on the queries that do not expect rotated and translated images in their answer sets), we cannot claim that the LCH is clearly better than the Harbin method.

Our experiments show that the Harbin approach is promising. The Harbin approach achieves better retrieval effectivness, in general, than the LCH and GCH methods. In particular, when answer sets expect rotated and translated images, Harbin can offer much better retrieval quality than either of the two traditional approaches. In contrast, using the LCH will eliminate most of the effectiveness. When answer sets do not include rotated and translated images, Harbin will generate equally good retrieval results as the LCH approach, and both can produce better results than the GCH approach. In practice, for a given query image, we should be able to find all similar images that may have different layouts; hence, if we are not concerned with retrieval efficiency, the Harbin approach should be a better choice. However, until now, the retrieval time has not been measured. We will discuss this issue next.

4.3 Retrieval Efficiency

The Harbin approach uses a high time complexity algorithm to search for the minimum cost matching (the Hungarian algorithm, which was described in Chapter 3, has the time complexity $O(mn^2)$). Therefore, it will not be practical to do a linear search in a large image collection using the Harbin approach. For example, with our platform, the average time for calculating the minimum cost matching in the Harbin approach, using 125

Queries	Harbin	LCH	GCH
cat	15.5	3.6	138.2
deer	14.5	20.1	26
feline	6.3	19.6	82.3
house	63.0	49.0	188.3
manhattan	51.4	11.6	39.7
oldboat	2.5	3.0	15.9
oldcar	13.9	6.7	10.1
pumpkin	59.3	20.6	77.9
racecar	10.5	22.5	64.4
soldier	9.5	27.7	44.5
swimming	1.1	2.1	2.0
wave	12.1	10.8	43.5
Average	21.6	16.4	61.0
Number of wins	6	6	0

Table 4.4: A comparison of three approaches for 12 queries in the second category using 125 colors, 8 by 8 blocks, and Δ=0.25

colors, 8 by 8 blocks, and a 0.25 Δ, is 0.016 seconds. When the database contains 20,000 images, one query using the Harbin approach will cost 320 seconds (without ranking images), which is unaffordable for most online applications. In Chapter 3, we introduced a method to refine the retrieval result of GCHs (which were obtained quickly) by re-ranking the top images using the Harbin approach, thus avoiding a linear search. For instance, when we utilize the Harbin approach using 125 colors, 8 by 8 blocks, and Δ=0.25, to re-rank the top 100 images, we will spend only 1.6 seconds on the refinement process. Figure 4.10 shows that after re-ranking the top 100 images, the Precision-Recall relationship is improved greatly. Surprisingly, the re-ranking approach takes much less time, but generates almost the same curve, as the Harbin approach—which is better than the GCH and LCH approaches. If we use only the GCH to do a linear search of our image database, it will spend 42.6 seconds to do one query (including the time to rank images). Therefore, with this method, which only adds a 3.72% overhead in query processing time, we obtain a much better retrieva1 quality. Nevertheless, in this thesis, we focus mainly on the effectiveness of image retrieval. The efficiency of the Harbin approach, or any database-oriented retrieval approach, is highly related to indexing structures and this is subject of further research.

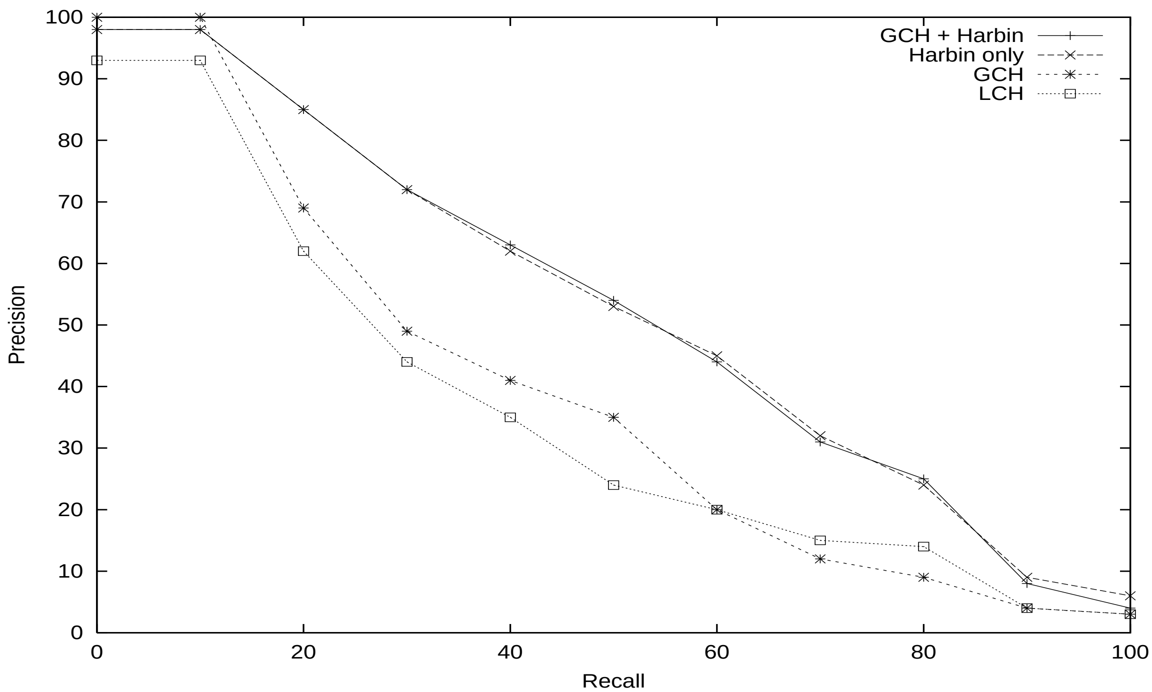


Figure 4.10: Re-ranking the top 100 images obtained by the GCH approach. All approaches, using 125 colors , 8 by 8 blocks, and Δ=0.25

Chapter 5

Conclusion and Future work

5.1 Conclusion

This thesis was motivated by the desire to overcome the shortcomings of two traditional color-based image retrieval approaches: the Global Color Histogram (GCH) which does not include spatial information; and the Local Color Histogram which cannot handle rotation and translation. We provided an overview of content-based image retrieval, investigated some techniques for color based image retrieval, and demonstrated the shortcomings of the two traditional approaches. We then described our new approach (Harbin) that models the problem of comparing the similarity between images in terms of finding minimum or maximum cost matchings in bipartite graphs. We investigated how to apply an heuristic threshold (Δ) to reduce noise caused by the edges connecting totally dissimilar blocks. The Harbin method employs high time complexity algorithms to compute the minimum or maximum cost matchings. However, this added time complexity is reduced by pruning the search space and applying the Harbin approach to rank only the top images returned by the GCH approach.

The experiments were carried out on a database of 20,009 images and 21 quey examples that can be categoried into two classes: the first included 9 query images that were similar to the images in their answer set by considering translation and rotation; and the second includes 12 query images that did not include rotated and translated images in their answer set. We evaluated the experimental results based on ε_{avg} and Precision-Recall relationships. The experimental results demonstrated that the Harbin approach yielded better retrieval effectiveness than the two traditional approaches in general situations. In particular, the Harbin approach showed its robustnees when dealing with query examples in the first catogory, whereas the LCH was ineffective. We studied the relationship between Δ and retrieval effectiveness, and showed that, through applying the Harbin approach to re-rank the top 100 images returned by the GCH approach, we gained both effectiveness and efficiency.

5.2 Future Work

This thesis presented a new method to compare similarities between images. The techniques may be further investigated in the following ways:

- We used the Euclidean function to calculate the distance between color histograms in RGB color space. Different distance functions, alternative color spaces, and other quantization schemes can be investigated.
- We simply partitioned images into equal-sized blocks. In the minimum or maximum matchings, some blocks of an image that belong to coherent regions may be separately matched to blocks of other images. Thus, some segmentation techniques to decompose images can be introduced to the Harbin approach.
- We have used the GCH approach as a filter to avoid a linear search. We believe that M-trees could be

introduced as the indexing structure for the Harbin approach, before using Δ. To improve query speed, indexing structure still needs to be further studied.

Bibliography

[Bie87] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review 94(2), 115-147*, 1987.

[Bim99] D. Bimbo. *Visual Information Retrieval*. Morgan Kaufmann, 1999.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. of the 1990 ACM Special Interest Group On Management Of Data (SIGMOD) Intl. Conf.*, pages 322–331, 1990.

[BVZ95] D. Bimbo, A. E. Vicario, and V. Zingoni. Symbolic description and visual querying of image sequences using spatiotemporal logic. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):609–621., 1995.

[Chi01] Vishal Chitkara. Color-based image retrieval using compact binary signatures. Master's thesis, University of Alberta, 2001.

[CJ90] S.-K. Chang and E Jungert. Pictorial data management based upon the theory of symbolic projection. *Journal of Visual Languages and Computing*, 2(3):195–215, 1990.

[CL84] S. Chang and S. Liu. Picture indexing and abstraction techniques for pictorial databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:475–484, 1984.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. of 23rd International Conference on Very Large Data Bases*, pages 426–435, Athens, Greece, 1997.

[CS96] J.L. Chen and G.C. Stockman. Indexing to 3d model aspects using 2d contour features. In *Proc. of Computer Vision and Pattern Recognition 96*, pages 913–920, 1996.

[CSY86] S. Chang, Q. Shi, and C. Yan. Iconic indexing by 2d strings. In *Proc. of the 1986 IEEE Computer Society Workshop on Visual Languages*, pages 12–21, Dallas, Texas, June 1986.

[Die97] R. Diestel. *Graph theory*. Graduate texts in mathematics. 173. New York : Springer, 1997.

[Dow93] J. Dowe. Content-based retrieval in multimedia imaging. In *Proc. of the SPIE Conference on Storage and Retrieval for Image and Video Databases (I)*, pages 164–167, San Jose, CA, 1993.

[DPS98] S. Dickinson, A. Pentland, and S. Stevenson. Viewpoint-invariant indexing for content-based image retrieval. In *Proc. of the IEEE International Workshop on Content-based Access of Image and Video Databases*, pages 20–30, Bombay, India, 1998.

[Ege91] M. Egenhofer. Reasoning about binary topological relations. In *Proc. of the 2nd International Symposium on Large Spatial Databases*, volume 525, pages 143–160, Berlin, 1991.

[EK72] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, April 1972.

[FBF+94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.

[FMBR95] S. Flank, P. Martin, A. Balogh, and J. Rothey. PhotoFile: A digital library for image retrieval. In *Proc. of International Conference on Multimedia Computing and Systems*, Washington, DC, May 1995. IEEE Computer Society.

[FT87] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[Gab82] H. N. Gabow. Scaling algorithms for network problems. In *Proc. of the 24th Annual Symposium on Foundations of Computer Science*, pages 248–258, Los Alamitos, Ca., USA, November 1982.

[GG98] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, June 1998.

[GJ97] A. Gupta and R. Jain. Visual information retrieval. *Communications of the ACM*, 40(5):70–79, 1997.

[GK94] T. Gevers and V.K. Kajcovski. Image segmentation by directed region subdivision. In *Proc. of the International Conference on Pattern Recognition 94*, pages A:342–346, 1994.

[GR95] V. N. Gudivada and V. V. Raghavan. Content-based image retrieval systems. *Computer*, 28(9), September 1995.

[Gra95] R. S. Gray. Content-based image retrieval: color and edges. Technical Report TR95-252, Dartmouth Institute for Advanced Graduate Studies, 1995.

[GW92] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, USA, 3rd edition, 1992.

[HCP95] W. Hsu, T. S. Chua, and H. K. Pung. An integrated color-spatial approach to content-based image retrieval. In *Proc. of the ACM Multimedia 95*, pages 305–313, 1995.

[HH94] M.W. Hansen and W.E. Higgins. Watershed-driven relaxation labeling for image segmentation. In *Proc. of International Conference on Image Processing (ICIP) III*, pages 460–464, 1994.

[HKM+97] J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *Proc. of the IEEE Computer Society Conference on Vision and Pattern Recognition*, pages 762–768, 1997.

[HMR96] T. Huang, S. Mehrotra, and K. Ramchandran. Multimedia analysis and retrieval system (MARS) project. In *Proc. of the 33rd Annual Clinic on Library Application of Data Processing - Digital Image Access and Retrieval*, University of Illinois at Urbana-Champaign, 1996.

[KMN98] L. M. Kaplan, R. Murenzi, and K. Namuduri. Fast texture database retrieval using extended fractal features. In *Proc. of the SPIE Conference on the Storage and Retrieval for Image and Video Databases VI*, pages 24–30, San Jose, California, January 1998.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quart.*, 2:83–97, 1955.

[LP96] F. Liu and R. W. Picard. Periodicity, directionality and randomness: Wold features for image modelling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):722–733, 1996.

[Lu99] G. Lu. *Multimedia Database Management Systems*. Artech House, 1999.

[LZC+94] X.Q. Li, Z.W. Zhao, H.D. Cheng, C.M. Huang, and R.W. H arris. A fuzzy logic approach to image segmentation. *International Conference on Pattern Recognition (ICPR)-A*, 94:337–341, 1994.

[MM96] B. Manjunath and W. Ma. Texture features for browsing and retrieval of image data. *the IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):37–42, August 1996.

[MP96] T. Minka and R. Picard. Interactive learning using a 'society of models. In *Proc. of the Computer Vision and Pattern Recognition (CVPR) 96*, pages 447–452, 1996.

[NBE+93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture and shape. Technical report, Almaden Research Center, San Jose, California, 1993.

[NHS84] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *the ACM Transactions on Database Systems*, 9(1):38–71, March 1984.

[PO93] E. Petrakis and S. Orphanoudakis. Methodology for the representation, indexing and retrieval of images by content. *Image and Vision Computing*, 11(8):504–521, 1993.

[PPS94] A. Pentland, Rosalind W. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *Proc. of the SPIE Conference on the Storage and Retrieval of Image and Video Databases II*, pages 34–47, San Jose, CA, 1994.

[PSTT93] G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora. Rotation invariant iconic indexing for image database retrieval. In *Proc. of the 7th International Conference on Image Analysis and Processing*, pages 271–278, Monopoli, Italy, 1993.

[PZM96] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *Proc. of the Fourth ACM Multimedia Conference*, pages 65–74, New York, NY, USA, November 1996.

[RHC99] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, April 1999.

[RSH96] Y. Rui, A. She, and T. Huang. Modified fourier descriptors for shape representation - a practical approach. In *Proc. of the First International Workshop on Image Databases and Multi Media Search*, 1996.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, June 1984.

[SC95] J. Smith and S. Chang. Automated image retrieval using color and texture. Technical Report CU/CTR 408-95-14, Columbia University, July 1995.

[SC96a] J. Smith and S. Chang. Tools and techniques for color image retrieval. In *Proc. of the SPIE conference on the Storage and Retrieval for Image and Video Databases IV*, pages 426–437, San Jose, CA, USA, 1996.

[SC96b] J. Smith and S. Chang. Visualseek: A fully automated content-based image query system. In *Proc. of the Fourth ACM International Multimedia Conference and Exhibition*, pages 87–98, Boston, MA, USA, November 1996.

[SC97] J. Smith and S. Chang. Visually searching the web for content. *IEEE Multimedia*, 4(3):12–20, 1997.

[SD96] M. A. Stricker and A. Dimai. Color indexing with weak spatial constraints. In *Proc. of the SPIE conference on the Storage and Retrieval for Image and Video Databases IV*, pages 29–40, San Diego, CA, USA, February 1996.

[SH91] M. J. Swain and D. H.Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[Smi97] J. Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Compression and Analysis*. PhD thesis, Graduate School of Arts and Sciences, Columbia University, San Diego, CA, USA, 1997.

[SNF00] Renato O. Stehling, Mario A. Nascimento, and Alexandre X. Falcao. On ”shapes” of colors for content-based image retrieval. In *The ACM Multimedia Conference*, Los Angeles, USA, Nov 2000.

[SO95] M. A. Stricker and M. Orengo. Similarity of color images. In *Proc. of the SPIE conference on the Storage and Retrieval for Image and Video Databases III*, pages 381–392, 1995.

[SSS00] M. Safar, C. Shahabi, and X. Sun. Image retrieval by shape: A comparative study. In *Proc. of the IEEE International Conference on Multimedia and Expo (I)*, pages 141–144, 2000.

[TMY78] H. Tamura, T. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Systems, Man, and Cybernetics*, 8:460–473, June 1978.

[YA94] L. Yang and F. Albregtsen. Fast computation of invariant geometric moments: A new method giving correct results. In *Proc. of the International Conference on Pattern Recognition (ICPR) 94*, pages A:201–204, 1994.

[YN99] B. Yates and R. Neto. *Modern Information Retrieval*. Addison Wesley, 1999.