2010

Retrieval of Spatially Similar Images using Quadtree-based Indexing

Naimul Khan
University of Windsor

RETRIEVAL OF SPATIALLY SIMILAR IMAGES USING QUADTREE-BASED INDEXING

by
Naimul Mefraz Khan

A Thesis
Submitted to the Faculty of Graduate Studies
through Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2010

RETRIEVAL OF SPATIALLY SIMILAR IMAGES USING QUADTREE-BASED INDEXING

by

Naimul Mefraz Khan

APPROVED BY:

Dr. Esam Abdel-Raheem
Electrical and Computer Engineering

Dr. Boubakeur Boufama
Computer Science

Dr. Imran Ahmad, Advisor
Computer Science

Dr. Robin Gras, Chair of Defense
Computer Science

April 28, 2010

Author’s Declaration of Originality

Abstract

Multimedia applications involving image retrieval demand fast response, which requires efficient database indexing. Generally, a two-level indexing scheme in an image database can help to reduce the search space against a given query image. The first level is required to significantly reduce the search space for the second-stage of comparisons and must be computationally efficient. It is also required to guarantee that no new false negatives may result. In this thesis, we propose a new image signature representation for the first level of a two-level image indexing scheme that is based on hierarchical decomposition of image space into spatial arrangement of image features (quadtrees). We also formally prove that the proposed signature representation scheme not only results in fewer number of matching signatures but also does not result in any new false negative. Further, the performance of the retrieval scheme with proposed signature representation is evaluated for various feature point detection algorithms.

Dedication

To my parents.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to Dr. Imran Ahmad, my supervisor, for his steady encouragement, patient guidance and enlightening discussions throughout my graduate studies. Without his help, the work presented here could not have been possible.

I also wish to express my appreciation to Dr. Boubakeur Boufama, School of Computer Science and Dr. Esam Abdel-Raheem, Department of Electrical and Computer Engineering for being in the committee and spending their valuable time and Dr. Robin Gras, School of Computer Science for serving as the chair of the defense.

Finally, I would like to thank all my friends on and off campus in Windsor for their consistent moral support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The old saying "*A picture is worth a thousand words*" [53] still bears strong importance in the modern world of science and technology. In fact, with the increasing use of streaming and embedded multimedia on the internet, the importance of pictures and videos is continuously on the rise. We see an extensive use of photographic materials in almost every step of our life, from buying a new car to monitoring heart conditions. Therefore, organization of photographic materials is a fundamental component of efficient operation of any multimedia system. Content-based image retrieval (CBIR) is any technique that helps to organize archive of digital pictures by their visual content for easy matching and retrieval [15].

One of the most challenging problems in the field of multimedia and computer vision is the automatic interpretation of image information. Generally speaking, exact characterization of human vision and perception is difficult, if not impossible. The knowledge base of humans gradually evolves with the help of the surroundings, whereas machines are usually built and trained for a specific task only. As a result, machines can only try to simulate human behavior. But obviously, the task of image retrieval from a large collection through human only is impractical. As a result, CBIR is of utmost importance and is a subject of intense research [15]. Generally speaking, CBIR requires automatic identification and understanding of the semantics of image contents for search and retrieval of similar images against a given query image.

One of the fundamental requirements of a CBIR system is the fast response which requires quick searching capabilities to search suitable images from a given image database. It is well known that the efficiency of search and retrieval of information in a database system is index dependent. In an image database with a large number of images, sequential search is not an option. Therefore, some indexing scheme is required to facilitate search and retrieval process. In an image database, the purpose of indexing is to reduce the cardinality of search space only. This allows the potentially similar images to go through more rigorous matching process that is generally more exhaustive and computationally expensive.

Most of the existing image indexing techniques are essentially based on quantifiable dominant visual image features such as color, texture, spatial arrangements of image contents, etc. [5, 40, 42, 47]. In [1], a new and different indexing scheme that is based on hierarchical decomposition of image space and quadtrees has been presented. In this scheme, abstract or symbolic images are used to improve the speed and efficiency of the search and retrieval process. A symbolic image is an abstraction of the actual or physical image and is based on its salient features [1]. Such images are generally used for comparisons only or to establish index structures to reduce the cardinality of the search space. The physical images are retrieved from the database only when a suitable match is determined. Some of the important advantages of this scheme are as follows:

- It is independent of basic geometric transformations (i.e., translation, rotation and scaling).
- The use of hierarchical decomposition allows representation of image at various level

of details (from coarse to fine) through the use of quadtrees.

- Its computational time is independent of the complexity of the physical image.
- It uses a two-level indexing that allows to curtail search space in the earlier stages of comparison with the help of *image signatures*.

One of the key aspects of this scheme is the improvement of retrieval efficiency by curtailing the search space in earlier stages of computation. An *image signature* in this scheme is essentially a representation of a symbolic image in a form so as to facilitate identification of only those images that satisfy a matching criteria. Therefore, selection of proper signature representation is extremely important. However, it involves the following trade-offs:

- The result of signature matching should be a limited number of potentially matching candidate images for the second level of filtering.
- The signature matching process should not result in any new *false negative*. In other words, due to signature matching, any potentially relevant image should not be left out from more intensive second level filtering.

These two competing and opposite trade-offs make it difficult to find a suitable signature representation scheme. In this thesis, We present a new image signature representation scheme. Like the scheme in [1], it also results in non-unique signatures but significantly increases the overall efficiency of the system by reducing the possible number of candidate images for intensive second level of filtering. In addition, it is mathematically proven that this scheme does not result in any new false negative.

Another important aspect of indexing scheme in [1] is the process through which symbolic images from the physical images are generated. Since physical images are not actually considered in the filtering process, the results and performance of the whole system is heavily dependent on accuracy of information encoded in symbolic images. In this scheme, a symbolic image is derived from feature points using corner detection algorithm. A survey of literature [56] indicates several corner detection methods. These methods vary considerably in term of their output and underlying philosophy. Therefore, in this thesis, We evaluate and compare results of three different popular corner detection algorithms, namely, the Smallest Univalue Segment Assimilation Nucleus (SUSAN) Detector [58], the Harris Detector [25] and the method proposed in [60], which we call the Wedge Detector.

The rest of this thesis is organized as follows: in Chapter 2, We provide an overview of Content-Based Image Retrieval. In Chapter 3, the idea of symbolic image representation is discussed along with some description of the three aforementioned corner detection methods. We also provide a brief description of the idea of hierarchical decomposition and introduce the notion of quadtree in this chapter. Chapter 4 provides details of the new signature representation scheme and various experimental results including extensive comparisons between the existing and the proposed signature representation schemes. Finally, Chapter 5 provides some concluding remarks.

Chapter 2

Content-based Image Retrieval

Content-Based Image Retrieval (CBIR) is an active area of research for quite some time [61]. The emphasis of CBIR is primarily on finding quantifiable visual features which can describe image contents. Prominent among these features are the image color, texture, shape, spatial image features etc [15]. Each of these features has its own advantages and limitations with some discussion in subsequent Sections. However, the ultimate choice of feature is generally domain specific.

Some techniques combine more than one of the above mentioned features and try to take advantage of the ones used. Query By Image Content, also known as QBIC by IBM [19] is a notable example of a such system. It combines color, texture and shape-based features. QBIC uses a graphical query language where the user can compose a query graphically by drawing, selecting or by some other graphical means. The images are stored in the database in such a way that they can be classified based on features such as colors, textures, shapes etc. The query is performed by matching the query image features with the database image features. VisualSEEK [57] is another notable scheme that allows the user to form visual queries by creating spatial arrangement of color regions through a diagraming process. It combines both color and spatial information to match a query image with the database images.

2.1 Color-based Retrieval

The most widely discussed visual feature to represent image contents is its color. A survey of literature indicates number of different ways of summarizing colors in an image [15]. Essentially, most of the techniques based on this feature make use of color histograms [19]. Any technique that is based on this feature alone does not take any advantage of either the local or the spatial image features. As a result, two entirely different images can be described by the same color contents [1]. Also, there are too many independent dimensions in a generic color histogram, such as choice of color space, choice of quantization in color space and quantization of the histogram values, which makes the interpretability between different color descriptors a difficult task [39]. A set of color and texture descriptors for MPEG-7 standard has been described in [39] which are well suited for both images and videos. One of them is the Scalable Color Descriptor (SCD), which uses the HSV color space and encodes the histograms using Haar transform. Another one is the Color Structure Descriptor (CSD), which uses the HMMD color space. However, for complex backgrounds or cases where the object occupies only a small portion of the scene, matching the histogram of the entire scene against the histogram of a model may fail to perform satisfactorily [26].

2.2 Texture-based Retrieval

Texture is another important feature for images that contain repetitive patterns such as carpets, grass field, checker boards etc. The use of texture features is generally domain specific with common application in aerial images. A discussion of texture feature extraction methods using wavelet and cosine transforms is presented in [16] and [38] . In [39], three different texture descriptors are listed for the MPEG-7 standard. One of them is the Texture Browsing Descriptor which concentrates on the regularity, directionality and coarseness of textures. The Homogeneous Texture Descriptor is probably the most robust texture descriptor specifically targeted to similarity-based retrieval [24]. It is calculated in the frequency domain by using different kinds of filters. The third one is the Edge Histogram Descriptor which captures the spatial distribution of edges. This descriptor is used in the cases where the texture is not homogeneous. As stated before, in most cases only color or texture features alone are not sufficient to describe the complete pictorial information present in an image.

2.3 Shape-based Retrieval

Shape-based retrieval [12, 63, 65] is another important aspect of image databases and a primary requirement in many applications, specially those dealing with security and surveillance. The principal focus of schemes using shape feature is on shape representation and on matching of different shapes. In [12], moment invariants and Zernike moments are used to represent the shape feature vectors. *k*-means clustering is then used to group the images with similar shapes together. To match a query image with database images, neural network is used with the clusters. Latecki, et al. [35] describes another method of shape representation using discrete curve evolution. Since contours of objects in an image can be distorted due to noise and segmentation errors, instead of using the original contours, an approximation is used. Every curve is approximated by a polygon by considering pair of consecutive line segments and substituting them by a single line segment by joining the end points. This iterative process is stopped with the help of a shape similarity measure, which is calculated by finding visual correspondence of the different parts. In [4], curves are represented by a set of segments. These features (curvature and orientation) are then arranged in a metric tree [13] to facilitate matching and retrieval. A Dynamic Programming (DP) approach has been discussed in [47]. In this approach, a shape is represented by a sequence of convex and concave segments. These segments are merged using DP where the program searches for the least cost path in a DP table. After merging, the merged segment is compared with larger segments from other shapes. All of the shape-based retrieval methods require extensive computations and are time-intensive and, hence, cannot be used when dealing with large collections of images [20].

2.4 Spatial Similarity-based Retrieval

Spatial arrangements of image objects and mutual relationships among them are the focus of spatial similarity-based retrieval schemes. Geographical and medical information domains are the primary beneficiaries of such type of image retrieval. Much of the approaches in spatial retrieval have been inspired by the classical *iconic indexing* scheme proposed by Chang, et al. [8].

2.4.1 Iconic Indexing

The basic idea of iconic indexing is to represent each object in an image by a symbol. A real image is then transformed to an equivalent symbolic image, where a matrix of symbols represent the objects of the real image. Figure 2.1 shows a sample image and the corresponding symbolic image representation. Here, the *vocabulary* or the set of symbols is

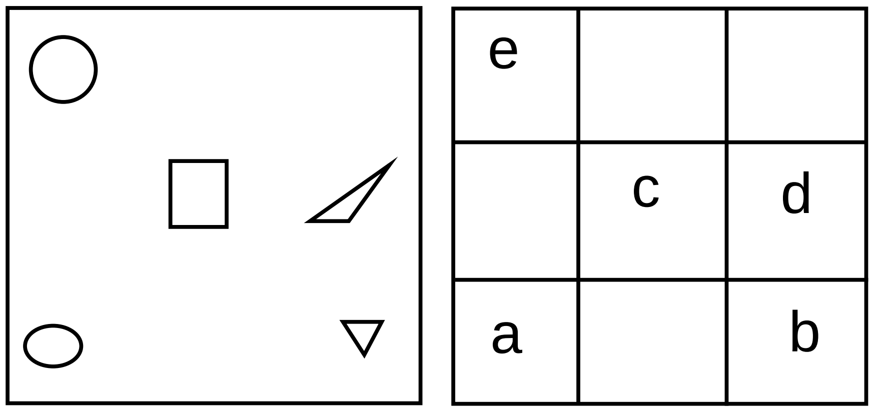


Figure 2.1: A sample image and the corresponding symbolic image

$V = \{a,b,c,d,e\}$. The symbols are projected in 2 directions (X and Y axes) to generate the 2D-string. Let us assume that the set of *operators* (explained in the following paragraph) is $U = \{=,<\}$. The 2D-string representation of the symbolic image for Figure 2.1 can be formulated as follows:

$$\begin{aligned} & (\quad a = e < c < b = d, a = b < c = d < e) \\ = \quad & (x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4\ x_5\ ,\ x_1\ z_1\ x_4\ z_2\ x_3\ z_3\ x_5\ z_4\ x_2), \end{aligned}$$

where

$$\begin{aligned} x_1\ x_2\ x_3\ x_4\ x_5 \ \ & is \ \ a\ e\ c\ b\ d \\ x_1\ x_4\ x_3\ x_5\ x_2 \ \ & is \ \ a\ b\ c\ d\ e \\ y_1\ y_2\ y_3\ y_4 \ \ & is \ \ = < < = \\ z_1\ z_2\ z_3\ z_4 \ \ & is \ \ = < = < \end{aligned} \tag{2.1}$$

Here, the operators $\{=, <\}$ denote spatial relationships. The operator $\{<\}$ denotes left-right or below-above spatial relationship for the first and second string respectively, whereas the operator $\{=\}$ denotes the "same location" spatial relationship.

In this way, an image can be represented by a 2D-string of symbols. A query image can then be transformed to a 2D-string in the same way where the user can specify the query graphically by drawing an iconic figure. This figure is then translated to a 2D-string and matched with the *iconic index* of a database image, which is basically the 2D-string representation of the database image itself. Therefore, the problem of image retrieval reduces to a 2D-string matching problem.

However, the 2D-string matching problem is similar to the Longest Common Subsequence Problem (LCS) [6] and is NP complete. Hence, this scheme is not computationally feasible for large image databases. Further, multiple occurrences of projection of symbols along the axes lead to ambiguity in the string representation. Moreover, since the information is encoded in the form of 2D-strings, this scheme fails to represent complete description of spatial relationships between the overlapping objects of an arbitrarily complex image [36]. This scheme also cannot recognize any rotational variant of an image [20].

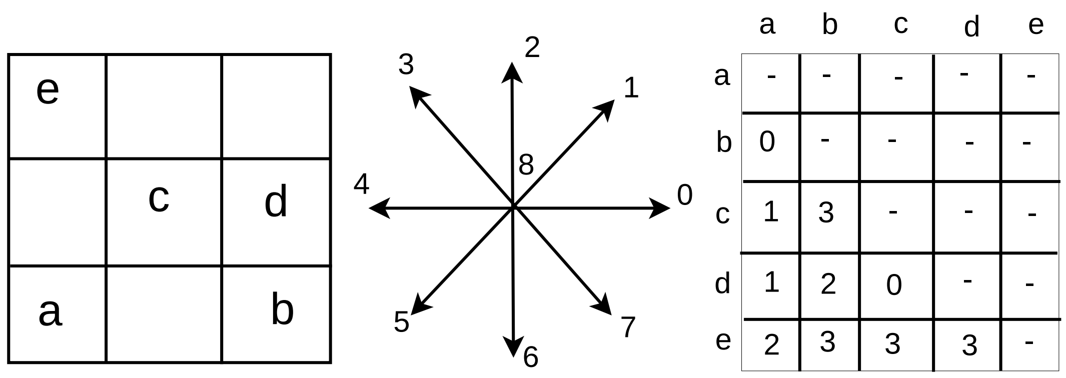
2.4.2 Extensions to Iconic Indexing

To overcome the aforementioned limitations, many extensions to the original iconic indexing scheme have been proposed. These extensions include extended 2D-strings [30, 31], 2D C and 2D C+-strings [28, 29, 37], 2D B-strings [36], $\Theta\mathfrak{R}$-strings [20] etc. In [30], the idea of splitting an object into several parts is introduced to handle overlapping objects. New local operators are also introduced to handle more spatial relationships. The 2D C-strings method described in [37] introduces 13 types of spatial relationships, which are basically modified and extended versions of the relationships described in [30] so that they can be incorporated in a global 2D-string representation. Another recent extension to the iconic indexing method is the “two dimension begin-end boundary string” or 2D-Be-string [62]. This scheme uses the idea of Minimum Bounding Rectangle (MBR) [44] to represent object locations inside an image more precisely. In this scheme, “dummy objects” are introduced to simplify the string representation, and the number of required operators are reduced to only one, the $\{=\}$ operator.

However, being variants of the LCS problem, most of these methods are computationally expensive. Also, most of them are not invariant to the basic geometric transformations (translation, rotation, scaling) [51].

In [45], a symbolic representation of images termed as *virtual image* is proposed and consists of entities (objects) and binary spatial relationships among them. Although virtual images are independent of translation and scaling, they are sensitive to rotation.

Also, most of the string-based techniques discussed above do not use any multi-dimensional indexing mechanism. As a result, for a given query image, an exhaustive search is made to find all relevant images which is not suitable for large image databases. Some string-based methods have also been proposed which use signature files for indexing. The method using Nine Direction Lower Triangular (9DLT) matrix was proposed in [7]. The 9DLT matrix basically uses 9 directional code values to represent binary spatial relationships among objects. An example of a 9DLT matrix is shown in Figure 2.2, where we show the 9DLT matrix for the symbolic image from Figure 2.1. Here, we can see that there are 9 possible directions in which a symbol (object) can exist with respect to another symbol. These 9 directions are represented by 9 values, $\{0,1,2,3,4,5,6,7,8\}$. Finally, the 9DLT matrix contains the values corresponding to the spatial relationships between the objects. Exten-



	a	b	c	d	e
a	-	-	-	-	-
b	0	-	-	-	-
c	1	3	-	-	-
d	1	2	0	-	-
e	2	3	3	3	-

Figure 2.2: Example of a 9DLT matrix

sions to the 9DLT matrix based method such as unique-ID based [11], generalized prime number based [9] and bit pattern based [10] matrix strategies are also found in literature. However, most of the matrix based approaches also suffer from considerable number of problems, such as storage overhead, inability to deal with geometrical transformations, etc. [22].

2.5 Multi-dimensional Indexing

To support efficient indexing and faster retrieval of images from a given image database, several multi-dimensional algorithms have also been proposed [5, 42, 46]. In [46], R-trees are used for indexing images represented as Attributed Relational Graphs (ARGs). In an ARG, image objects are represented by the graph vertices and their relationships are represented by the edges. Both the vertices and the edges are labeled by attributes corresponding to properties of the objects and their mutual relationships, respectively. The ARGs corresponding to the images are mapped to a high-dimensional point using FastMap [18]. These high-dimensional points are then indexed in the database using R-trees [23]. R*-trees [2] for image indexing has been discussed in [42], which are improvement over the

R-trees. Graph-based image representation and indexing algorithm has been proposed in [5]. Several other multi-dimensional indexing methods such as multi-dimensional binary-trees [3], K-D-B-trees [52], BD-trees [14], G-trees [34], grids [43], multi-attribute hashing [54] etc. also exist. However, unlike quadtrees [55], none of these indexing methods can be directly related to the notion of recursive decomposition of an image (explained in Section 3.2), which is one of the key motivations behind our retrieval system because of its ability to represent an image at various level of details. As a result, quadtree is picked as the preferred representation method of multi-dimensional indexing in our case. The structure and formal definitions related to quadtrees are described in detail in Section 3.2.2.

2.6 Summary

This chapter discusses the evolution of content-based image retrieval throughout the past few years. The various areas of CBIR have been presented with emphasis on spatial similarity-based retrieval. From the discussions we see that despite having advantages and applications in their own domains, most of the existing methods suffer from several drawbacks.

Chapter 3

Symbolic Image Representation and Recursive Decomposition

The key to the notion of a spatial similarity-based image retrieval system is the symbolic or abstract image representation of an actual image. A symbolic image is an abstraction of the actual image and contains pertinent information necessary for comparison purposes only. Such a representation also provides physical data independence, and hence, eliminates the repetitive task of spatial and semantic information encoding [1].

3.1 Feature Image Representation

Each image I_i in an image database is unique and distinct and can contain one or more objects $O_k, k \geq 1$. In many applications, an image can be described by the help of a set of significant feature components, defined as $F_k = \{F_k^1, F_k^2, \ldots, \mathrm{F}_k^{r_k}\}$. These features can be described as follows [21]:

Physical features: Physical features are those features which can be expressed quantitatively. These features can be measured directly from the physical image. As a result, physical feature identification can be an automated process. Example of physical features are number of image objects, object positions, object areas etc.

Logical features: Logical features are those features which can not be calculated directly, rather those have to be retrieved semantically. Attaching a label (e.g. "sky") to a part of the image is an example of a logical feature.

Features can also be described as:

Global features: These kind of features are directly associated with physical features and can be easily computed. These features are based on the overall image composition and generally applicable to the whole image. Example of these kind of features are image size, overall image color composition etc.

Local features: Local features contain low-level characteristic information associated with the image. Computation of these kind of features require extensive low level processing, and hence, are computationally expensive. These kind of features usually contain spatial information about the image. Image boundary points and centroids of image objects are example of these kind of features.

In spatial similarity-based retrieval, Only those image features that provide spatial information about the image objects (spatial positions, mutual relationships etc.) are considered important. These features are termed as *spatial features*.

A spatial feature F_k^j of an image object O_k in a 2D image space can be represented as a set of points $P_k^j = \{p_k^{j,1}, p_k^{j,2}, \ldots\}$, where $p_k^{j,m} = (x_k^{j,m}, y_k^{j,m})$ are the coordinates of the said point. These points can be tagged with labels from the representative domain to capture necessary semantics [1]. Each of these individual points represent some spatial feature of an image object and is called a *feature point*. For the sake of clarity, it is assumed that only a single feature point represents a spatial feature of an image object. Thus, the entire image is represented by a set of representative feature points.

The identification and labeling of feature points in a physical image essentially converts it into an equivalent symbolic image, also called *feature image* [1]. For feature image extraction, one can take advantage of the existing image processing techniques such as detection of corner points. Corner points represent important local features in images. Generally speaking, they are the points that have high curvature and lie at the junction of different brightness regions of an image. Generally corner points are not affected by illumination and have the property of rotational invariance [59]. Therefore, corner points can serve as feature points for generation of an equivalent symbolic image of an actual image. As indicated before, symbolic images are used for comparison and to determine suitability of match for retrieval of actual images against a given query image. Actual images are retrieved from the database only when the retrieval results are presented to the user. Therefore, feature image generation is one of the fundamental steps in the retrieval process and can effect the overall accuracy and performance of the system.

A survey of literature indicates several different corner point detection methods [59]. The output provided by these methods vary considerably due to differences in the underlying techniques of each method. Most of the commonly used methods usually rely on intensity derivatives to find high curvature points. Some methods use direct intensity values of the pixels inside a pre-defined window to find corners whereas some other methods try to define a corner model and match the image regions with it. As a result, the methods behave differently in terms of accuracy, robustness and sensitivity [60].

To test our retrieval system against varying detection algorithms, we selected the following three important contemporary methods. Our selection is essentially based on the differences in their underlying philosophies and, consequently, different output feature images in terms of spatial similarity.

3.1.1 SUSAN Detector

The method introduced in [58] is a popular corner detector. In this method, no image derivatives are used and no noise reduction is needed. The method works on a small pre-defined window called Univalue Segment Assimilating Nucleus (USAN). A pixel is considered as the *nucleus* and the area of USAN, also called mask, is defined by the pixels which have same or similar intensity as that of the nucleus. Let r_0 and r represent coordinates of nucleus and some other point within the mask $c(r, r_0)$ of an area M. Every pixel is compared to the the nucleus using the comparison function:

$$c(r, r_0) = \exp\left\{ -\frac{[I(r) - I(r_0)]^6}{t} \right\} \tag{3.1}$$

where t is the threshold and determines how similar the intensity values should be so as to be considered in the same USAN. This threshold varies from image to image between different image databases and must be tuned for each image database. The size of USAN is given as:

$$n(M) = \sum_{r \in M} c(r, r_0) \tag{3.2}$$

If c is the rectangular function then n is the number of pixels in the mask which are within t of the nucleus. The area of USAN falls to a local minima for a corner (Smallest

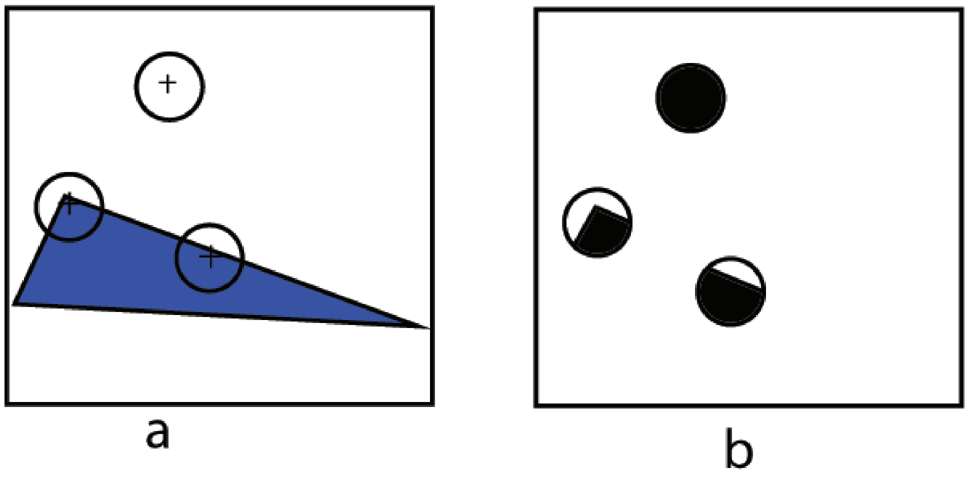


Figure 3.1: The principle behind SUSAN (a) Pre-defined windows with nuclei marked (b) Area of USAN falling to minimum in case of a corner

USAN). The idea is evident from Figure 3.1, where we see an image containing a simple triangular object. Three sample pre-defined windows are shown in Figure 3.1-a , with the nucleus (center) of each marked. In Figure 3.1-b, the area of USAN for each of the three windows is marked in black. As stated before, the pixels with same or similar intensity are considered to be in the same USAN as the nucleus. We see that for the corner, the area of USAN falls to minimum indeed.

The response of SUSAN operator is given by:

$$n(M) = \begin{cases} g - n(M), & \text{if } n(M) < g \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

where g is the geometric threshold which determines the acute level of a corner. The value of g determines the minimum size of the univalue segment. This geometric threshold is necessary to filter out the false positives which can result from detecting multiple corners on a straight edge. This threshold ensures that a corner is acute enough to be considered as

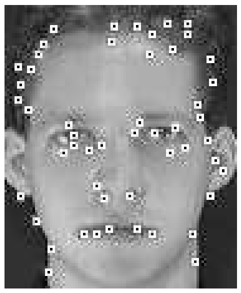


Figure 3.2: Application of the SUSAN detector on a facial image

a corner point.

Figure 3.2 shows the output corner points marked for a facial image after applying the SUSAN detector. The image is taken from the ORL face database [64], details of which can be found in Section 4.4. The threshold t was tuned properly to generate the best output. Here, we see that the SUSAN principle gives reasonably correct output in terms of detected corners. However, we also see that some expected corners (specially around the chin area) were not detected. If the intensity values are not different enough for the surroundings of the corner, the area of USAN will not fall to a local minima and, as a result, the SUSAN detector will not detect a corner.

3.1.2 Harris Detector

Harris detector [25] is another common corner detection method. In this method, the image intensity derivatives are used to find corners with a pre-requisite of a smoothing operator to reduce the sensitivity to noise. The Harris detector is an improvement over Moravec's corner detector [41]. In Moravec's detector, a small window is considered and the average intensity value in the window is calculated. Then the window is shifted in different directions to calculate the average change in intensity. For a corner, the shift of the window in any direction will result in large change of intensity value. As the shifts are too discrete and the used window is rectangular, the output of the Moravec's detector has room for improvement. Harris proposed a different way of calculating corner scores to get rid of these drawbacks. For an image pixel (x,y) of an image I, the Harris method first calculates the *autocorrelation* matrix given as:

$$M(x,y) = S * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{3.4}$$

where I_x and I_y are the image intensity gradients and S denotes the smoothing operator [49]. The use of a smoothing operator is optional, but advisable as this function uses image derivatives which can be susceptible to noise. The greatest eigenvalue of $M(x,y)$ corresponds to the rate of change in the direction of highest variation, while the smallest eigenvalue corresponds to the rate of change in perpendicular direction of the highest variation. If both of these eigenvalues have high magnitudes, then (x,y) can be considered as a corner point. Since calculating eigenvalues is computationally expensive, the following approximation has been suggested to determine the corner strength:

$$R(x,y) = det(M(x,y)) - \kappa(trace(M(x,y)))^2. \tag{3.5}$$

Suitable value of κ has been determined to be $0.04 - 0.15$ [56]. The value of $R(x,y)$ is generally larger than the specific threshold for a corner point. The threshold is again, dependent on the type of images used and may vary from database to database.

Figure 3.3 shows the output corner points marked for the same facial image as in Figure 3.2 after application of the Harris detector. The threshold was properly tuned again to generate acceptable output. Here, we see that compared to the output of the SUSAN detector, the corner points detected by the Harris detector are spread more equally across the

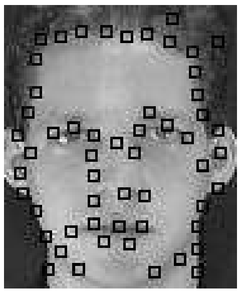


Figure 3.3: Application of the Harris detector on a facial image

image. Rather than using the pixel intensity values directly, the Harris detector uses image derivatives. As a result, the output provided is different than that of the SUSAN detector.

3.1.3 Wedge Detector

We call the method described in [60] as the Wedge detector. It uses a simple corner model to detect corner shapes. The corner model consists of a wedge (the corner), having its origin at the center of a circular neighborhood (the background) and is described by an angular position θ and an angular width φ (Figure 3.4). For each pixel, the circular area around it is segmented into background and foreground by using a sigmoid function. The corner strength is then calculated using the following equation:

$$C(x,y,W_\theta^\varphi) = \int\int_{C_{x,y}} |W_\theta^\varphi(i,j) - sig(I(x+i,y+j) - \bar{I}(x,y))| di dj, \tag{3.6}$$

where $C_{x,y}$ is the circular area around (x,y), W_θ^φ is the binary map of the corner model being used, $sig()$ is the sigmoid function for background and foreground segmentation and $\bar{I}(x,y)$ is the average pixel intensity in $C_{x,y}$. The sigmoid function $sig()$ is defined as follows:

$$sig(I(x,y) - \bar{I}(x,y)) = \frac{1}{1 + e^{s(I(x,y) - \bar{I}(x,y))}}. \tag{3.7}$$

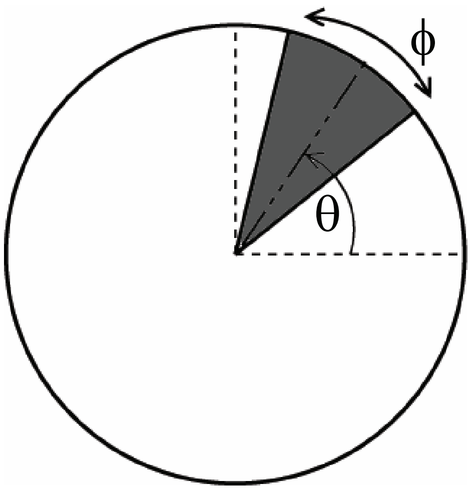


Figure 3.4: A simple corner model

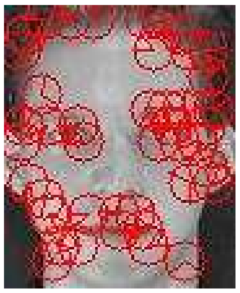


Figure 3.5: Application of Wedge detector on a facial image

Here, s is a constant whose value is adjusted to determine to which extent a pixel is segmented as a foreground pixel. Instead of using a constant threshold for background and foreground segmentation, use of a sigmoid function is deemed to be more stable.

Instead of finding the optimal θ and φ for the corner model which can be too time consuming, the values are approximated by calculating the union of smaller foreground wedges. Finally, only those points are considered as corners for which the value of $C(x,y,W_\theta^\varphi)$ from Equation 3.6 exceeds a pre-determined corner strength value. This threshold is the primary control parameter and varies from one database to another.

Figure 3.5 shows the output corner points marked for the same facial image as in Figure

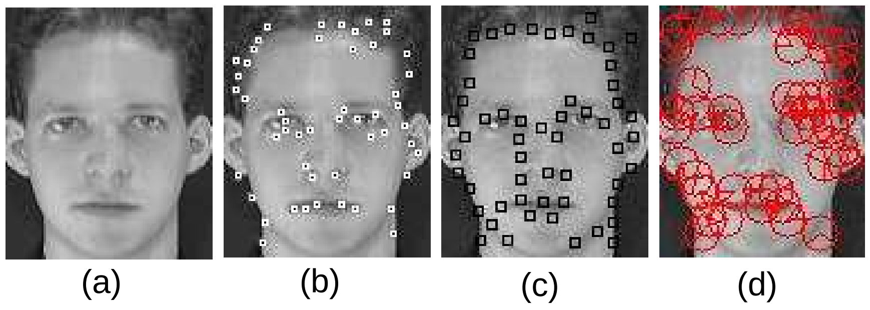


Figure 3.6: (a) Input image (b) Output of the SUSAN Detector (c) Output of the Harris Detector (d) Output of the Wedge Detector

3.2 and 3.3 after applying the Wedge detector. The corner strength threshold was properly tuned to generate acceptable output. Here, we see that the output of this method has similarity to the SUSAN detector. Like SUSAN detector, this method also does not detect expected corner points around the chin region. The reason for the similar outputs from these two methods is the similarity in their underlying mechanism. SUSAN detector also relies on a simple segmentation to obtain the USANs. But unlike SUSAN, this method explicitly compares the segments to a pre-established corner model. In that sense, this method has some difference in principle compared to the SUSAN detector and, hence, the output corner points are not exactly the same.

Figure 3.6 shows the comparison of the results obtained by these three corner detection methods on the same image (Figure 3.6-a) as depicted before. As can be seen, the corner points obtained by these three methods are considerably different in terms of spatial relationships among them. As a result, our retrieval system will behave differently to each of these methods and we have observed the same in experiments (Section 4.4).

3.2 Recursive Decomposition

Recursive decomposition is the method of dividing the feature image into distinct regions to capture and recognize the spatial and semantic relationships among individual feature points and is defined as follows [1]:

Definition 3.1. The process of recursively dividing an image space into four equal size quadrants $0, \ldots, 3$ is termed as the *recursive decomposition* of an image. The resultant quadrants are recognized by four directional relations as North-West (NW), North-East (NE), South-West (SW), South-East (SE), respectively. The decomposition process stops only when each and every feature point can be identified by a distinct quadrant and, therefore, each quadrant can contain exactly one point in it.

The decomposition hierarchy can be mapped to a top-down built quadtree [55] of height h such that:

- The root of the tree is assumed to be at level $l = 0$ and represents complete image.
- The level $i, i \geq 0$, of the tree corresponds to the ith level of decomposition.
- The leaf nodes of the tree represent the smallest quadrants resulting from the decomposition, and each encloses only a single feature point.

Figure 3.7 is an example of a hierarchically decomposed feature image and its corresponding quadtree representation. In this figure, the root of the tree corresponds to the original non-decomposed image. Each level of the quadtree corresponds to subsequent levels of decomposition. The circles represent internal nodes of the tree whereas leaf nodes correspond to the smallest quadrants. Quadrants containing feature points are represented by black rectangles and white rectangles represent empty quadrants.

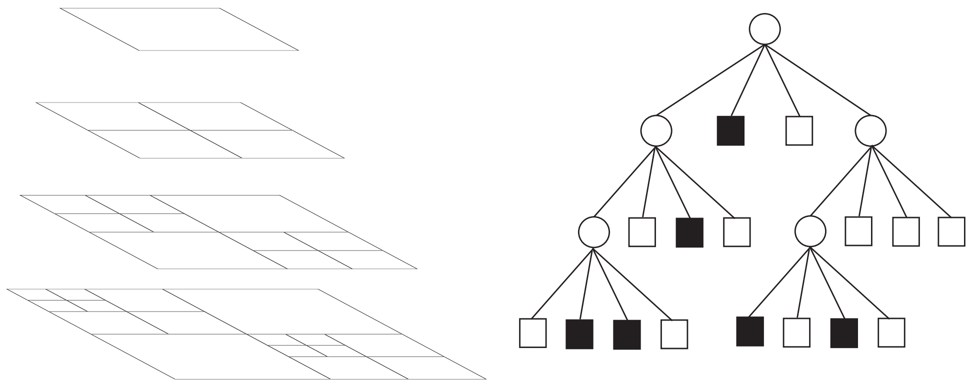


Figure 3.7: A recursively decomposed image and its corresponding quadtree

3.2.1 Geometric Transformation Independence

During the semantic conversion process, each feature point is considered as a reference point to recalculate the orientation of the principal axes of the feature image using the method described in [27]. The basic idea here is to fit a simple straight line

$$y = a + bx$$

to the set of feature points based on the criterion of least absolute deviations [50]. Based on the fitted line, the principal axes and the centroid of the image are recalculated. After all the feature points are considered and the new centroid and principal axes of the feature image are found, the coordinates of all the feature points are recalculated with respect to the new centroid and principal axes (Figure 3.8), thus making the representation translation, rotation and scale independent.

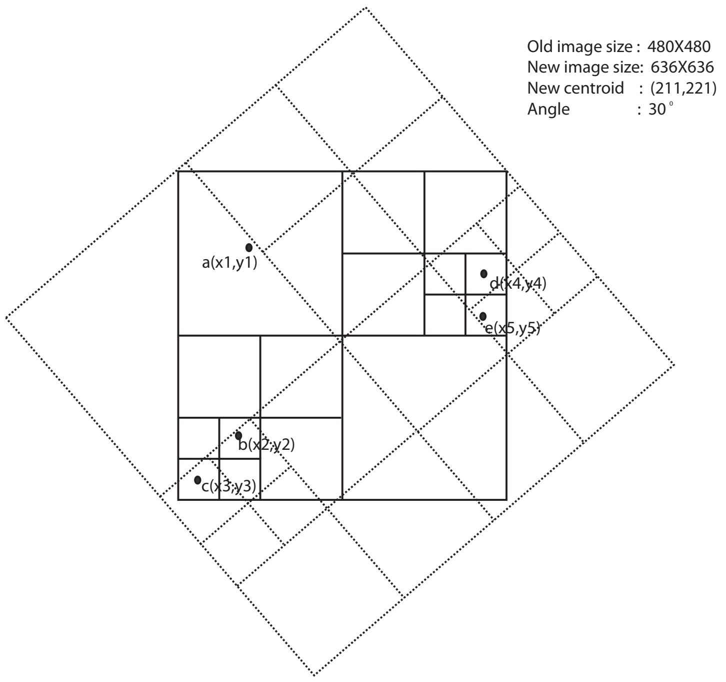


Figure 3.8: A feature image with recomputed boundaries parallel to the new principal axis

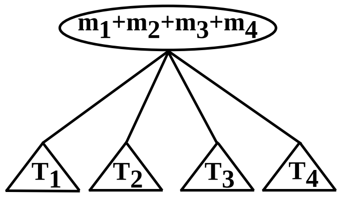
3.2.2 Quadtrees

To establish a measure of similarity between the images matched in first stage of filtering, their corresponding quadtrees are matched. The quadtrees are matched with the help of a distance function (Definition 3.7). The distance function is defined in such a way that the distance is computed node by node, starting from the root and going down along both of the trees gradually. This results in reduced number of comparisons since it allows to eliminate trees that appear to be different at the initial stages of processing. To formulate spatial filtering and indexing scheme, the following terminologies are necessary [1]:

Definition 3.2. A quadtree is defined as follows:

- A single node $\textcircled{M}$ is a quadtree. This node is a *leaf* as well as the root of the tree. $m \geq 0$ is called the *occupancy* of this node. It is meant to capture the number of feature points in the region of the image corresponding to the node. The *coordinate sequence* of this node is λ, the empty sequence.

- If T_1, T_2, T_3, T_4 are quadtrees whose roots have occupancies m_1, m_2, m_3 and m_4 respectively, where $m_1 + m_2 + m_3 + m_4 \geq 0$, we have that



is a quadtree. The node $(m_1 + m_2 + m_3 + m_4)$ is the root of the resulting quadtree. The leaves of T_1, T_2, T_3 and T_4 are the leaves of the resulting quadtree. The coordinate sequence of root node is λ. If *seq* is coordinate sequence of a node in T_j, for $1 \leq j \leq 4$, then $j \bullet seq$ is the coordinate sequence of this node in the resulting quadtree, where $\bullet$ is the sequence concatenation operator.

In these definitions, the root node of the quadtree T is denoted by root(T) and the occupancy of the node n of a quadtree is denoted by occupancy(n).

Definition 3.3. The level of a node, n, in a quadtree is the length of the coordinate sequence of that node and is denoted by level(n).

Definition 3.4. The height of a quadtree T, denoted by height((T), is one more than the maximum level of any node in the quadtree.

Definition 3.5. The ith *approximation* of a quadtree T, for $i \geq 0$, is the quadtree which results by removing all nodes on level $j > i$. It is denoted by $T^{(i)}$.

Definition 3.6. A quadtree is *complete* iff each leaf node has an occupancy of 0 or 1.

Now the distance function is defined as:

Definition 3.7. The distance between two quadtrees T and U, $d(T,U)$ is defined as follows:

case 1: Suppose height(T) = height(U) = 1 and occupancy(root(T)) + occupancy(root(U)) = 0. Then

$$d(T,U) = 0$$

case 2: Suppose height(T) = height(U) = 1 and occupancy(root(T)) + occupancy(root(U)) > 0.

Also, let M =occupancy(root(T)) and N = occupancy(root(U)). Then

$$d(T,U) = \frac{|M-N|}{max(M,N)}$$

case 3: Suppose height(T) = 1 and occupancy(root(T)) = 0 and height(U) > 1. Then

$$d(T,U) = 1$$

case 4: Suppose height(T) = 1, occupancy(root(T)) = 1, height(U) > 1 and at least one child of the root node of U has an occupancy greater than 0. Then

$$d(T,U) = \frac{|N-1|}{N}$$

case 5: Suppose height(T) = 1, occupancy(root(T)) = 1, height(U) > 1 and at least one child of the root node of U has an occupancy equal to 0. Then

$$d(T,U) = 1$$

case 6: Suppose height(T) > 1 and height(U) > 1.

For $1 \leq j \leq 4$, let the subtree of T and U determined by the nodes having coordinate sequence j be called T_j and U_j respectively.

Let occupancy(root(T)) $= M$ and occupancy(root(U)) $= N$.

For $1 \leq j \leq 4$, let occupancy(root(T_j)) $= m_j$ and occupancy(root(U_j)) $= n_j$. Then

$$d(T,U) = \max\left(\sum_{j=1}^{4} \frac{m_j}{M} d(T_j,U_j), \sum_{j=1}^{4} \frac{n_j}{N} d(T_j,U_j)\right)$$

Theorem 3.1. *Let T and U be two complete quadtrees. Then, for $i \geq 0$,*

$$d(T^i,U^i) \leq d(T^{(i+1)},U^{(i+1)})$$

Proof. (Adapted from [1])

Basis − Case $i = 0$**:** *Case 1:* height(T) = height(U) = 1.

Then, $T^{(0)} = T^{(1)}$ and $U^{(0)} = U^{(1)}$ and the result follows.

Case 2: height(T) $= 1$ and occupancy(root(T)) $= 0$ and height(U) > 1.

Then, $T^{(0)} = T^{(1)} = T$ and height($T^{(1)}$) $= 1$.

From Definition 3.7, Case 3 we have that

$d(T^{(0)},U^{(0)}) = d(T^{(1)},U^{(1)}) = 1$.

Case 3: height(T) $= 1$ and occupancy(root(T)) $= 1$ and height(U) > 1.

Then, $T^{(0)} = T^{(1)} = T$ and height($T^{(1)}$) $= 1$.

Suppose occupancy(root($U^{(0)}$))$= N$. From Definition 3.7, Case 4 we have that

$d(T^{(0)}, U^{(0)}) = |N-1|/N$.

Now, height($T^{(1)}$) $= 1$. $\therefore$ from Definition 3.7, Case 4 and 5 we have that

$d(T^{(1)}, U^{(1)}) \geq |N-1|/N$ and the result follows.

Case 4: height(T) > 1 and height(U) > 1.

For $1 \leq j \leq 4$, let the subtree of T and U determined by the nodes having coordinate sequence j be called T_j and U_j respectively. Let occupancy(root(T)) $= M$ and occupancy(root(U)) $= N$.

For $1 \leq j \leq 4$, let occupancy(root(T_j)) $= m_j$ and occupancy(root(U_j)) $= n_j$.

Without loss of generality, we assume that $M \geq N$. By Definition 3.7, Case 2 we have that

$d(T^{(0)}, U^{(0)}) = (M-N)/M$.

Now, consider the following term:

$$Q = \sum_{j=1}^{4} \frac{m_j}{M} \frac{|m_j - n_j|}{max(m_j, n_j)},$$

where, if $m_j = n_j = 0$, the jth summand is equal to 0.

For $1 \leq d \leq 4$, if $m_d \geq n_d$, then

$$\frac{m_d}{M} \frac{|m_d - n_d|}{max(m_d, n_d)} = \frac{m_d - n_d}{M}.$$

Again, if $m_d < n_d$, then

$$\frac{m_d}{M} \frac{|m_d - n_d|}{max(m_d, n_d)} > \frac{m_d}{M} \frac{m_d - n_d}{m_d} = \frac{m_d - n_d}{M}.$$

$$\therefore\ Q \geq \sum_{j=1}^{4} \frac{|m_j - n_j|}{max(m_j, n_j)} = \frac{(M-N)}{M}$$

. Now, $d(T^{(1)}, U^{(1)}) \geq Q$.

$$\therefore\ d(T^{(1)}, U^{(1)}) \geq Q \geq \frac{(M-N)}{M} = d(T^{(0)}, U^{(0)}).$$

Hence, the result follows.

Induction: Assuming that the theorem is true for $i = 0, \ldots, k$, we will show that it is true for $i = k+1$.

For $1 \leq j \leq 4$, let the subtree of T and U determined by the nodes having coordinate sequence j be called T_j and U_j respectively.Let occupancy(root(T)) $= M$ and occupancy(root(U)) $= N$.

For $1 \leq j \leq 4$, let occupancy(root(T_j)) $= m_j$ and occupancy(root(U_j)) $= n_j$.

Then, we have that the subtree of $T^{(k)}$ and $U^{(k)}$ determined by the nodes having coordinate sequence j are $T_j^{(k-1)}$ and $U_j^{(k-1)}$ respectively. We also have that:

$$d(T_j^{(k)}, U_j^{(k)}) = \max\left(\sum_{j=1}^{4} \frac{m_j}{M} d(T_j^{(k-1)}, U_j^{(k-1)}), \sum_{j=1}^{4} \frac{n_j}{N} d(T_j^{(k-1)}, U_j^{(k-1)})\right),$$

and

$$d(T_j^{(k+1)}, U_j^{(k+1)}) = \max\left(\sum_{j=1}^{4} \frac{m_j}{M} d(T_j^{(k)}, U_j^{(k)}), \sum_{j=1}^{4} \frac{n_j}{N} d(T_j^{(k)}, U_j^{(k)})\right).$$

But, by hypothesis we already have that $d(T_j^{(k-1)}, U_j^{(k-1)}) \leq d(T_j^{(k)}, U_j^{(k)})$. Hence, our result follows.

□

This theorem implies that for two trees T and U, if $d(T^i, U^i) \geq \gamma$, where $\gamma \geq 0$ is the allowable deviation in similarity or *threshold* and $i \geq 0$, then the distance between these two trees at subsequent levels $d(T^{(i+1)}, U^{(i+1)})$ will never be less than the threshold γ. This means that one can start comparing the trees from the root level and as soon as it is discovered that the distance between the two trees at a given approximation is greater than the threshold, its corresponding database image can be discarded without any additional computation. The distance function is defined in such a way (Definition 3.7) that the minimum distance between two trees can be 0 and the maximum distance can be 1. As a result, the value of γ can range from 0 to 1.

A suitable value of γ is critical to the filtering process. A small value though reduces the search space, but carries the risk of leaving out some of the potential candidates. On the other hand, a value close to 1 may not leave out potential candidates but may result in a larger set of candidates for a potential match, including those which may not be close to the query image at all. The optimal value is both application and user expectation dependent and cannot be predicted in advance.

With the help of a quadtree, we can represent an image at various levels of details and provide different measurements of similarity between a query and the database images. In terms of quadtree representation, the extent of fineness is directly proportional to the depth of the quadtree [1]. The smaller the depth of the tree, the more coarse or less accurate is the description. The most coarse is the root level, where all feature points are only in one square. Hence, the *less* the depth of the tree traversed during comparisons, the *more* is the chance that there will be a match, specially when most of the images in the database are similar to some extent. This is an important observation and serves as the basis of our new signature representation scheme.

3.3 Summary

In this chapter, the notion of symbolic image representation has been presented. The importance of underlying corner detection method in our proposed system has been discussed with details of three different corner detectors: SUSAN, Harris and the Wedge detector. The idea of recursive decomposition with its advantages and the process of quadtree building and matching are also presented.

Chapter 4

The Proposed Signature Scheme with Experimental Results

As mentioned before, the first of the two-level hierarchical indexing scheme in [1] is based on the concept of signature matching. It serves as a spatial filter to discredit unqualified images against a given query image. The second stage (as described in Section 3.2.2) is based on tree matching that not only serves as a secondary filter but also provides a measure of similarity to rank-order retrieved images. The main purpose of two-staged filtering is to reduce the cardinality of the search space in first level of filtering so that the second level that performs more intensive computations has to deal with only a limited set of candidate images. It is important to note that in the first stage of filtering, if the search space is too tight and small, there is a likelihood of leaving out some of the potentially useful matches a.k.a *false negatives*. On the other hand, if the search space is too wide, there is a possibility of retrieving irrelevant information a.k.a *false positives* [17]. In an ideal system, the number of both false negatives and false positives should be zero. In reality, it is always better to have false positives than false negatives and most of the image retrieval systems are designed on this philosophy.

4.1 The Existing Scheme

In [1], a signature is defined as a 32-bit number and is a combination of both disjoint and superimposed coding methods [17, 48]. Each signature in this scheme consists of two disjoint fields. First one of these fields is based on population standard deviation of the number of feature points in the four quadrants after only first level of decomposition. It can be represented by the following equation [1]:

$$S_1^i = \sqrt{\frac{\sum_{j=0}^{3}(a_j - \mu)^2}{\sum_{j=0}^{3} a_j}}, \tag{4.1}$$

where $\mu = \sum_{j=0}^{3} a_j/4$ and a_j is the number of feature points in each of the four quadrants after the first level of decomposition.

The second field is the average number of feature points at each level of decomposition of the image and can be represented by the following equation [1]:

$$S_2^i = \frac{\sum_{j=0}^{3} a_j}{h}, \tag{4.2}$$

where h is the height of the quadtree.

After building the signature, the notion of *Tolerance Factor* (Definition 4.1) is used to relax the signature to accommodate the concept of similarity. A step-by-step example of how a signature is formed is given later in Section 4.3.

The main problem with this signature representation is the generalization in both of its fields. As a result, this signature matching scheme results in too many false positives. This can be observed from the results in Section 4.4. As stated before, having too many false positives leads to poor performance due to higher number of comparisons. Moreover, image retrieval systems are most likely to be used in specialized applications where images are generally similar to each other to some extent (e.g. medical imaging, facial recognition, etc.). As an example, a typical facial image database consists of close-up facial shots of all subjects. Here the basic shape of face in all of the images is similar. When most of the images in a database have some type of similarity, the distribution of feature points in the first level of decomposition does not exhibit much variance for creation of distinguished signatures. As a result, many of the signatures will be identical. The signature representation scheme defined in [1] only deals with the standard deviation in first level and average number of points in each level. This is not suitable for many applications, particularly those involving specialized databases. However, it is important to note that whatever can be accepted in the second stage of filtering *must not* be rejected in the first stage.

4.2 The Proposed Scheme

Our new signature representation scheme is based on the observations and tradeoffs stated before. The proposed scheme results in significantly less number of qualified signatures for second level of filtering and, hence, results in significantly improved system performance. Also, our signature representation is entirely based on the idea of recursive decomposition and quadtrees and, hence, does not incur any additional computational or storage overhead.

Our signature representation scheme is based on the important observation that in most of the images, the coordinate locations of feature points are mostly *coarse*. This is due to the fact that usually the feature points occur at locations where there is a sharp intensity or illumination change. This occurs around the object boundaries whereas most of the

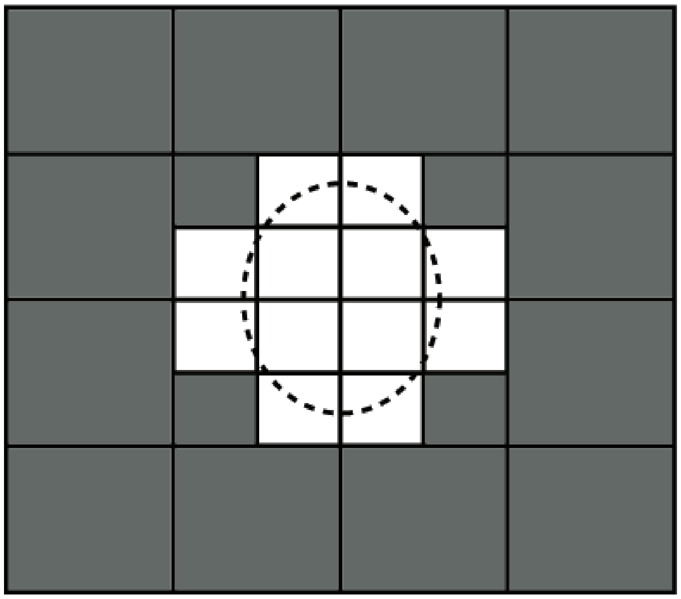


Figure 4.1: Distribution of feature points in a typical image

images in real applications contain only few objects. Even if an image contains a lot of objects, compared to the pixel resolution of the image, the number of feature points and, hence, the number and placement of these points are still coarse. Therefore, during recursive decomposition, if we only consider the information available to us in the first level of decomposition, we may not gain much useful information and if the two images have some degree of similarity, they may yield the same signatures. This concept is illustrated in Figure 4.1 where the coordinate locations of the feature points are denoted by dots appearing in an oval shape in the center of the image. Here we can see that a number of quadrants at different levels of decomposition are empty, i.e., they do not contain any feature point. To be more precise, before *level 3*, the feature points are all concentrated only in one quadrant at each level of decomposition. In this figure, shaded areas indicate quadrants with no feature points.

From this observation, we can intuitively assume that the quadrants with no feature points should not be involved in subsequent computations. Later we formally prove that discarding these empty quadrants does not result in loss of relevant information and hence, cannot generate any new false negative during the quadtree matching process. Based on these observations, the new signature representation scheme [32] is given as follows:

- As before, a signature S of a quadtree T of an image is a 32-bit number. However, in new scheme, the signature S consists of four 8-bit disjoint fields S_i, $1 \leq i \leq 4$. Each of these fields corresponds to one of the four quadrants in the recursive decomposition.
- if height$(T) = 1$, then $S = 0$.
- if height$(T) = 2$, then $S_i = \%$ of total number of feature points belonging to the quadrant i.
- if height$(T) > 2$, for $1 \leq i \leq 4$ let the subtree of T determined by the nodes having coordinate sequence i be called T_i. Also, each S_i is further divided into four equal size disjoint fields (i.e. 2 bits each) $S_{(i,j)}$, $1 \leq j \leq 4$ such that each field corresponds to one of the four quadrants in the recursive decomposition of T_i. Then for each T_i:
    1. if height(T_i) $= 1$, then $S_i = 0$.
    2. if height(T_i) $= 2$, then $S_{(i,j)} = \%$ of total number of feature points belonging to the quadrant corresponding to j.
    3. if height(T_i) > 2:
        - If at least two of the four immediate descendants of T_i have occupancy greater than 0, then $S_{(i,j)} = \%$ of total number of feature points belonging to the quadrant corresponding to j.

  - In all other cases, $T_i = T_{(i,j)}$, such that root($T_{(i,j)}$) $=$ only immediate descendant of root(T_i) with non-zero occupancy. Go to step 1.

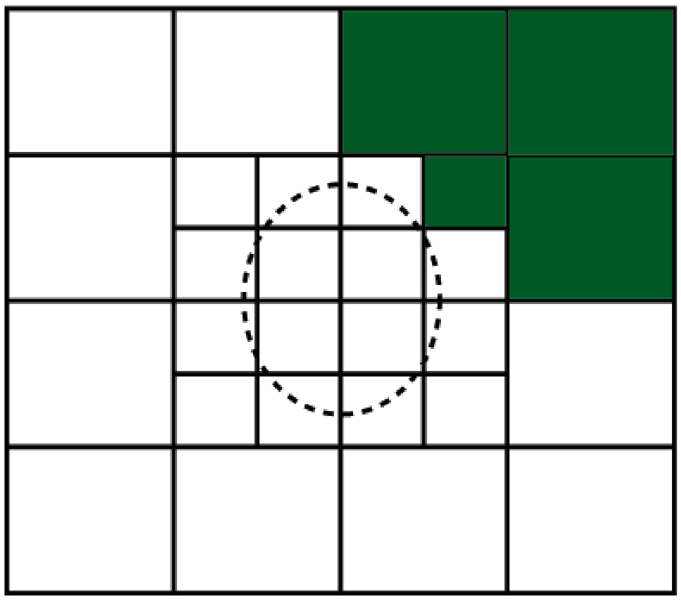
In simple words, the representation implies that we can keep on recursively subdividing each of the quadrants until we get at least two sub-quadrants, each containing at least one feature point in it. This will allow us to discard most of the empty quadrants. Figure 4.2 shows the application of this method for the top right quadrant of Figure 4.1. Here we can observe that at level 3, there are three sub-quadrants with non-zero occupancy. Therefore, we stop at level 3 and build the signature accordingly.

In this scheme, we need to represent % of total number of feature points with the help of only two bits. This is achieved by representing the % value(p) with the help of 4 ranges:

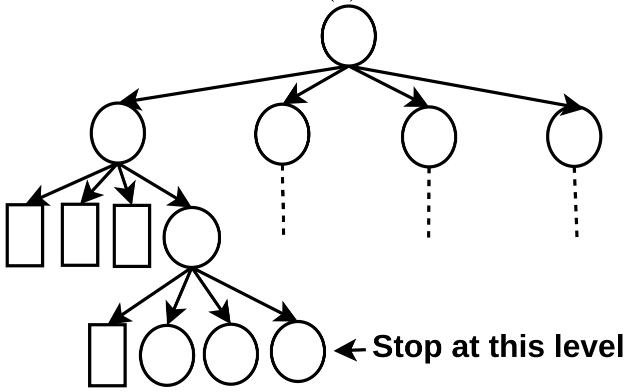
- $p < 25\%$
- $25\% \leq p < 50\%$
- $50\% \leq p < 75\%$
- $p > 75\%$

Although this may generalize the information to some extent but the flexibility in the generated signatures out weights this generalization. Further, the generated signatures provide significant improvement in the overall system performance as can be observed from the experimental results.

It should be noted that since this is the first stage of filtering, if any image that can be matched with the query image in the second stage is left out in the signature matching phase, it will become a false negative. In our signature representation, whenever we traverse a level down in the quadtree to find nodes with non-zero occupancy, we discard empty nodes. If



(a)



(b)

Figure 4.2: New signature representation method (a) A sample recursively divided image and (b) its corresponding quadtree

we can prove that even after discarding these nodes, the tree distance still remains the same as before (or less), we can guarantee that our signature representation scheme will not result in any new false negative. The proof here is given only for one of the four subtrees after first level of decomposition but others are similar. Before first level of decomposition, the signature is trivial and does not result in any new false negative.

Theorem 4.1. *If there exists two trees T and U such that no node of T or U has more than one immediate descendant with non-zero occupancy up to level m and level n respectively, then $d(T,U) \geq d(T_p,U_q)$, where T_p and U_q are two subtrees such that root(T_p) = only descendant of root(T) with non-zero occupancy at level m and root(U_q) = only descendant of root(U) with non-zero occupancy at level n.*

Proof. Without loss of generality, we can assume that height$(T) > 1$ and height$(U) > 1$.

Case 1: $m = n$. From Definition 3.7, Case 6 we have that:

$$d(T,U) = \max\left(\sum_{j=1}^{4} \frac{m_j}{M} d(T_j,U_j), \sum_{j=1}^{4} \frac{n_j}{N} d(T_j,U_j)\right)$$

where T_j and U_j are the subtree of T and U determined by the nodes having coordinate sequence j respectively. occupancy(root(T)) $= M$ and occupancy(root(U)) $= N$. For $1 \leq j \leq 4$, occupancy(root(T_j)) $= m_j$ and occupancy (root(U_j)) $= n_j$.

Now, let T_a and U_b are two subtrees such that:

root(T_a) = only immediate descendant of root(T) with non-zero occupancy and

root(U_b) = only immediate descendant of root(U) with non-zero occupancy.

then:

- occupancy(root(T_j)) = M when $j = a$

- occupancy(root(T_j)) = 0 otherwise

Similarly:

- occupancy(root(U_j)) = N when $j = b$
- occupancy(root(U_j)) = 0 otherwise

Hence,

$$
\begin{aligned}
&d(T,U)\\
&\quad = \max\left(\sum_{j=1}^{4} \frac{m_j}{M} d(T_j,U_j), \sum_{j=1}^{4} \frac{n_j}{N} d(T_j,U_j)\right)\\
&\quad = \max\left(\frac{M}{M} d(T_a,U_a), \frac{N}{N} d(T_b,U_b)\right)\\
&\quad = \max\left(d(T_a,U_a), d(T_b,U_b)\right)
\end{aligned}
$$

- if $a = b$, then we have $d(T,U) = d(T_a,U_b)$
- if $a \neq b$,then for $d(T_a,U_a)$ we have occupancy (root(U_a)) = 0.

  Also, height(U_a) = 1 and height(T_a) $>$ 1, and from Definition 3.7, Case 3 we have:

  $d(T_a,U_a)$ = 1.

  Similarly, $d(T_b,U_b)$=1.

  Therefore, in this case $d(T,U) = 1 \geq d(T_a,U_b)$

  $\Rightarrow d(T,U) \geq d(T_a,U_b)$

  Similarly $d(T_a,U_b) \geq d(T_c,U_d)$, where T_c and U_d are two subtrees such that:

  root(T_c) $=$ only immediate descendant of root(T_a) with non-zero occupancy.

  root(U_d) $=$ only immediate descendant of root(U_b) with non-zero occupancy.

  $\therefore d(T,U) \geq d(T_a,U_b) \geq d(T_c,U_d) \geq \ldots \geq d(T_p,U_q) \Rightarrow d(T,U) \geq d(T_p,U_q)$.

Case 2: $m \neq n$. This Case is similar to Case 1 above. Let us assume that T_m and U_n are two subtrees such that:

root(T_m) = last descendant of root(T) having only one immediate descendant with nonzero occupancy and root(U_n) = last descendant of root(U) having only one immediate descendant with nonzero occupancy.

By Case 1 it has already been proven that $d(T,U) \geq d(T_m,U_n)$. Now, let T_a and U_b are two subtrees such that:

root(T_a) = only immediate descendant of root(T_m) with non-zero occupancy and root(U_b) = only immediate descendant of root(U_n) with non-zero occupancy.

Then we exactly have Case 1 from which rest of the proof follows.

$\therefore d(T,U) \geq d(T_p,U_q)$.

□

This theorem is very important for us since it ensures that our signature representation will not result in any new false negative. It is important to note that even if theoretically one may find a representation scheme to generate a unique signature for each database image but in reality, the cost of finding such a representation may not be worth it and may significantly degrade overall system performance. Even though this scheme generates non-unique signatures, it is quite simple and does not incur any additional overhead. All of the quadtree information required here is already available for use in second level of filtering.

4.3 Tolerance Factor

Another important aspect of signature representation is the *Tolerance Factor* (Definition 4.1). Tolerance Factor is essential to accommodate the concept of similarity between a database and the query image [1] and effects the number of matched images retrieved. Therefore, it is obvious that it has an immediate effect on the number of retrieved signatures. The higher the value of the Tolerance Factor, the higher the number of retrieved signatures and possibly higher number of matched images. In our representation, the modified definition of Tolerance Factor is given as:

Definition 4.1. The Tolerance Factor (TF) is an addition of $\pm x$ to the % values of $S_{(i,j)}$, $1 \leq i, j \leq 4$ for each S_i of a database image signature S for similarity-based search and retrievals by providing a range-search capability [1].

TF $= 0$ is a special case and provides an exact match. The value of x in TF depends on application and extent of similarity. This allows us to test a range of image signatures for a possible match against a given query image signature S_q, computed without any tolerance. In the second stage of filtering, only those images are taken into consideration for which the binary AND operation between the query image signature S_q and database image signature S results in the query image signature S_q i.e. $S_q \cap S \rightarrow S_q$.

As an example, suppose we have assigned the following 2 bit binary codes to the 4 ranges:

- $p < 25\% \rightarrow 00$
- $25\% \leq p < 50\% \rightarrow 01$
- $50\% \leq p < 75\% \rightarrow 10$

- $p \geq 75\% \rightarrow 11$

Now, let us assume $S_{(1,1)}$ is 43% and accordingly the assigned binary code is 01. Let us also assume that the value of x for calculating TF is 10. Then we have two range values and consequent binary codes for $S_{(1,1)}$:

- $S'_{(1,1)} = S_{(1,1)} - x = 33 \rightarrow 01$
- $S''_{(1,1)} = S_{(1,1)} + x = 53 \rightarrow 10$

Our final representation of $S_{(1,1)}$ is computed by taking binary representations of the three values and performing a binary OR operation. In other words:

$$S_{(1,1)} \rightarrow S_{(1,1)} \cup S'_{(1,1)} \cup S''_{(1,1)}$$

$$\Rightarrow S_{(1,1)} \rightarrow 01 \cup 01 \cup 10$$

$$\Rightarrow S_{(1,1)} \rightarrow 11.$$

Similarly, lets assume $S_{(1,2)} \rightarrow 00$, $S_{(1,3)} \rightarrow 01$ and $S_{(1,4)} \rightarrow 10$. Then, by concatenating these binary codes we find S_1:

$$S_1 \rightarrow S_{(1,1)} | S_{(1,2)} | S_{(1,3)} | S_{(1,4)}$$

$$\Rightarrow S_1 \rightarrow 11 \mid 00 \mid 01 \mid 10$$

$$\Rightarrow S_1 \rightarrow 11000110.$$

Similarly, lets assume $S_2 \rightarrow 00111000$, $S_3 \rightarrow 00101001$ and $S_4 \rightarrow 00111100$. Finally, by concatenating these codes, we find the 32-bit signature S for a database image:

$$S \rightarrow S_1|S_2|S_3|S_4$$

$$\Rightarrow S \rightarrow 11000110\ 00111000\ 00101001\ 00111100.$$

Now, suppose our query image signature is $S_q \rightarrow 10000110\ 00101000\ 00100001\ 00110100$. Then, by performing a binary AND operation between S and S_q, we find:

$$\begin{array}{c} 11000110\ 00111000\ 00101001\ 00111100 \\ \cap \\ 10000110\ 00101000\ 00100001\ 00110100 \\ \hline 10000110\ 00101000\ 00100001\ 00110100 \end{array}$$

which is equal to the query image signature S_q. Hence, this database image will be accepted for the second stage.

4.4 Experimental Results

To verify the presented concepts, we conducted a series of experiments on three separate image databases with each database concentrating on different aspects of the system (average computational time, effect of changing threshold value etc.).

4.4.1 Databases and Experimental Setup

The first of these databases is the popular Olivetti-Oracle Research Lab (ORL) face database [64]. The ORL face database consists of 400 frontal face images; images of 40 individuals with 10 variations of each in terms of pose, illumination, facial expression (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). The second database is a binary shape database. It consists of binary images of closed shapes of different ob-

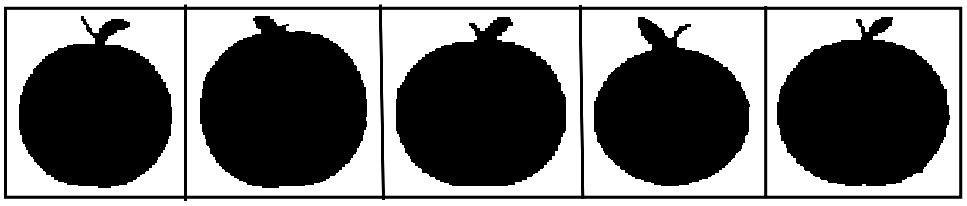


Figure 4.3: Spatially similar images from the binary shape database

jects collected from [33]. We randomly selected 2000 different shape objects. In order to demonstrate geometrical variance independence, we generated five geometric variants of each original image. As a result the database contains 12000 images with exactly six instances of each image. The above mentioned two databases vary from each other considerably in terms of their complexity and the number of images. The images from the ORL face database are generally more complex in terms of illumination and intensity variation than the images from the binary shape database. As a result, it is more difficult to produce consistent symbolic images from the facial images. On the other hand, the binary shape database has significantly higher number of images. These two databases are used to compare the performance of the proposed system to the existing system since the main purpose of our proposed system is to reduce the number of matched signatures in the first stage and consequently the overall CPU time.

The third image database, called spatial database is a small database and consists of binary images of spatially similar objects collected from [33]. Images of only 20 objects are collected such that each object has 5 spatially similar images, making it a database of only 100 total images. The experiments on this database are conducted mainly to observe the effect of changing the value of distance threshold γ, as described in Section 3.2.2.

Figure 4.3 shows the spatially similar images of an apple from this database. Some important characteristics of the image databases used are summarized in Table 4.1.

For the first two databases, each image is used as a query image and compared against all other database images. Recorded results are the the average results (number of retrieved images, computational time, etc.) over all instances of execution. For the spatial database, we randomly selected one of the five spatially similar images of each object as a query image and the results are averaged for all of the 20 objects.

Each of the corner point detection methods described in Section 3.1 has a key control parameter. We adjusted this parameter to a suitable value for each of the three methods by visually observing the output for few randomly selected representative images from each database.

The experiments are repeated for a range of TF values. To match the TF of both systems for comparisons, we chose $TF \in \{0.0, 0.1, 0.2\}$ for the existing system whereas for the proposed system, we chose $TF \in \{0, 5, 10\}$. An interval of 5 in the proposed system is considered to be equivalent to an interval of 0.1 in the existing system and is a reasonable choice for the proposed system as $0 \leq S_{(i,j)} < 100$ for all $S_{(i,j)}$. The distance threshold value γ is fixed at 0.5 to discard unqualified images for the first two databases. For the spatial database, it is varied from 0 to 1 with a step size of 0.2.

Characteristic	ORL Face Database	Binary Shape Database	Spatial Database
Number of Total Images	400	12000	100
Number of Original Images	400	2000	100
Rotational Variants Per Image	0	2	0
Scale Variants Per Image	0	2	0
Translation Variants Per Image	0	1	0
Minimum Image Size (pixels)	92 X 112	52 X 50	75 X 42
Maximum Image Size (pixels)	92 X 112	609 X 492	124 X 158

Table 4.1: Summary of characteristics of image databases

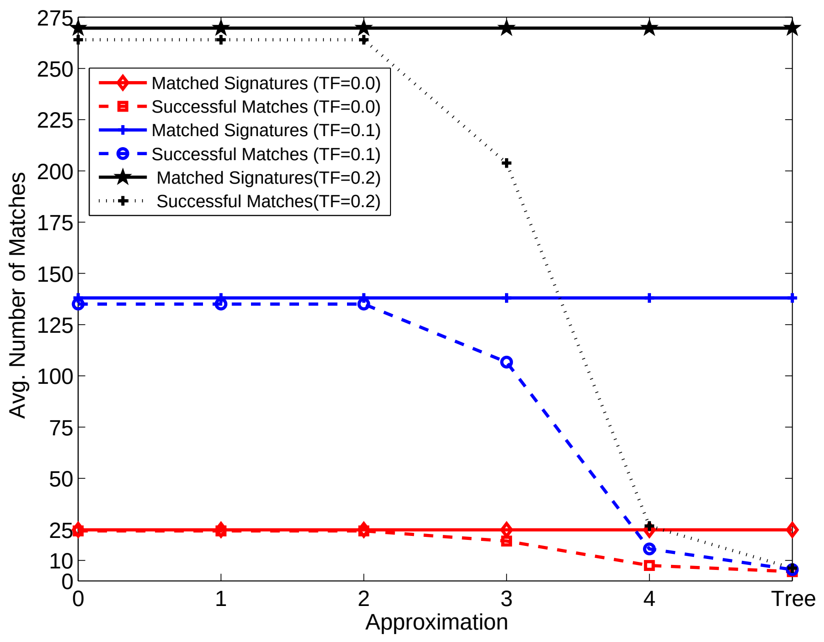
4.4.2 Results and Analysis

All of the experimental results are collected using same hardware configuration for both the existing and the proposed system for all experiments − a personal computer with an Intel Core^{TM}2 Quad processor running at a speed of 2.33GHZ and containing 6 GB of RAM whereas the developed application is in Java.

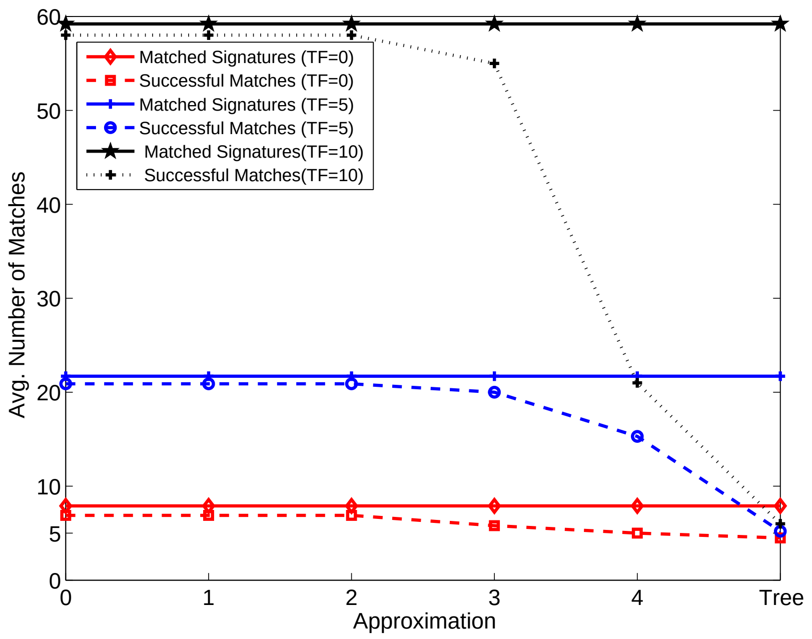
Figures (4.4-4.9) show the results of comparison of average number of retrieved and matched signatures for the first two databases using the three corner detection methods. In these figures, the label “Tree” indicates the entire tree matching or the n^{th} approximation. From these figures, we can observe that for all cases, the proposed signature representation scheme results in considerably fewer number of matched signatures. As a result, the overall performance of the system is improved, which is evident from the CPU time comparisons presented in Figure 4.10. The solid lines in Figure 4.10 represent CPU times for the existing method while the dashed/dotted lines represent CPU times for the proposed method.

In binary shape database, there are exactly six known instances of each image. Ideally for $TF = 0.0$, only six signature should match with the query image signatures and, as a result, we should retrieve only six images. However, it is important to note that since image signatures are not unique, the system may retrieve more than six matching signatures. However, as can be observed from Figures (4.7-4.9), the new method results in significantly reduced number of matched signatures, making it nearly optimal when $TF = 0.0$.

Also, for the ORL face database although all of the 10 images for a particular individual have similar shapes, they may not be spatially similar to one another. The concept of spatial similarity in this domain translates to pose rather than shape as one might expect. As a result, our system is not expected to retrieve all of the 10 images of a particular individual but may retrieve images that have similar pose, even if they belong to different individual.

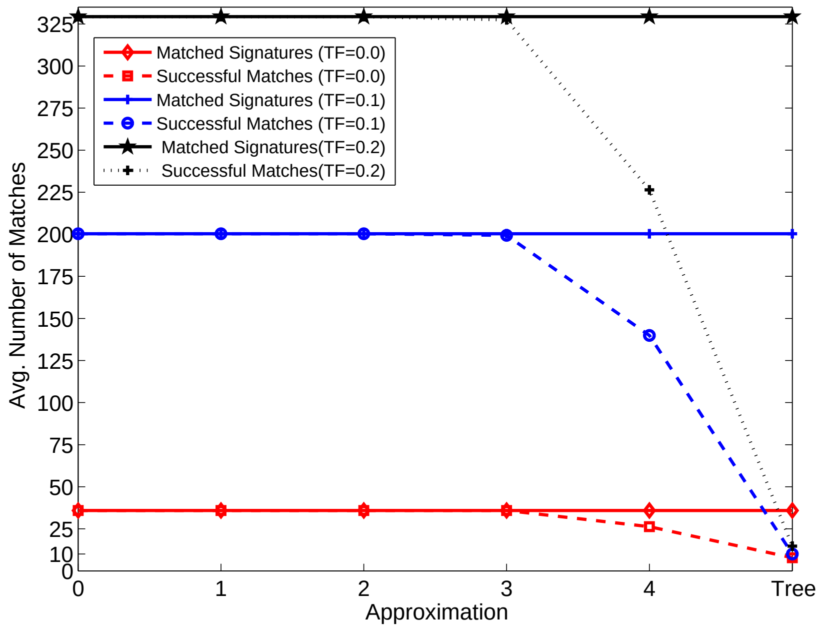


(a) Existing Method

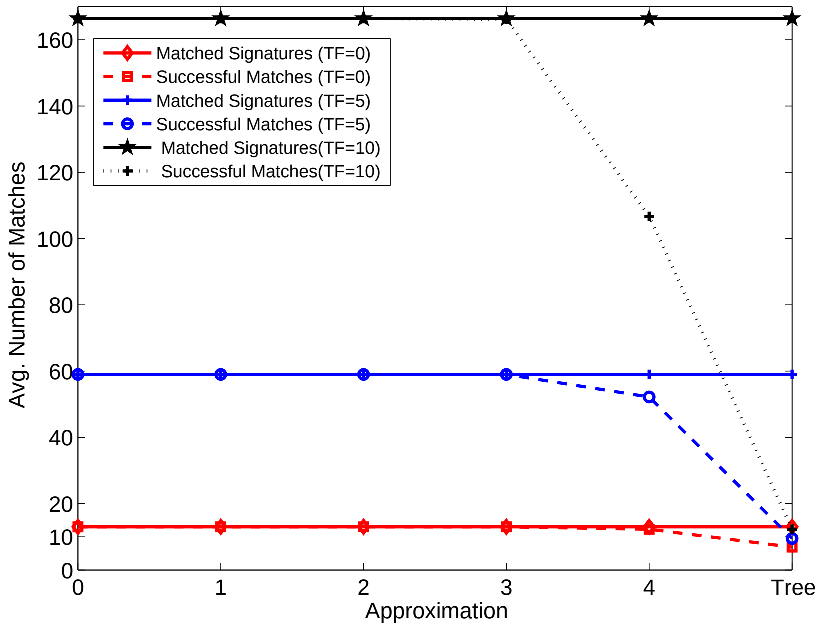


(b) Proposed Method

Figure 4.4: Comparison of results for the ORL face database using SUSAN

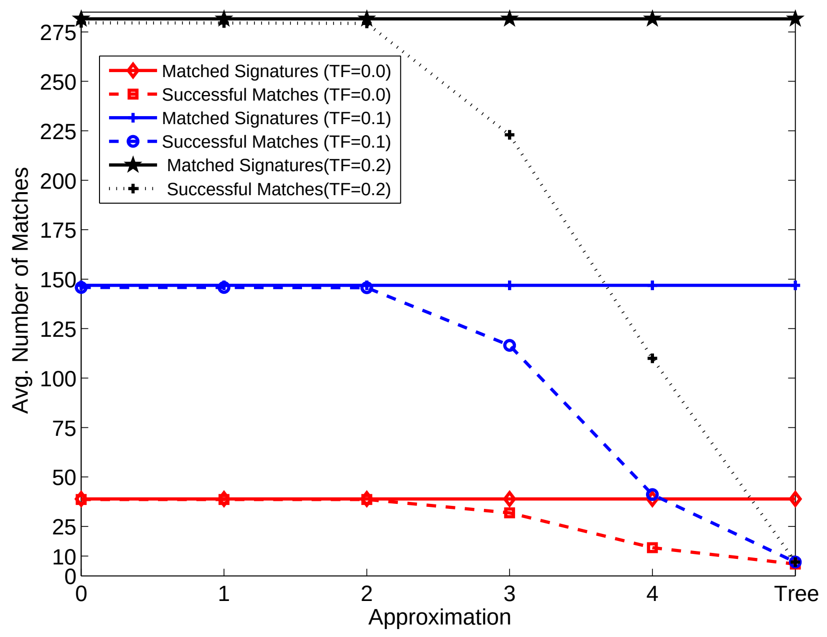


(a) Existing Method

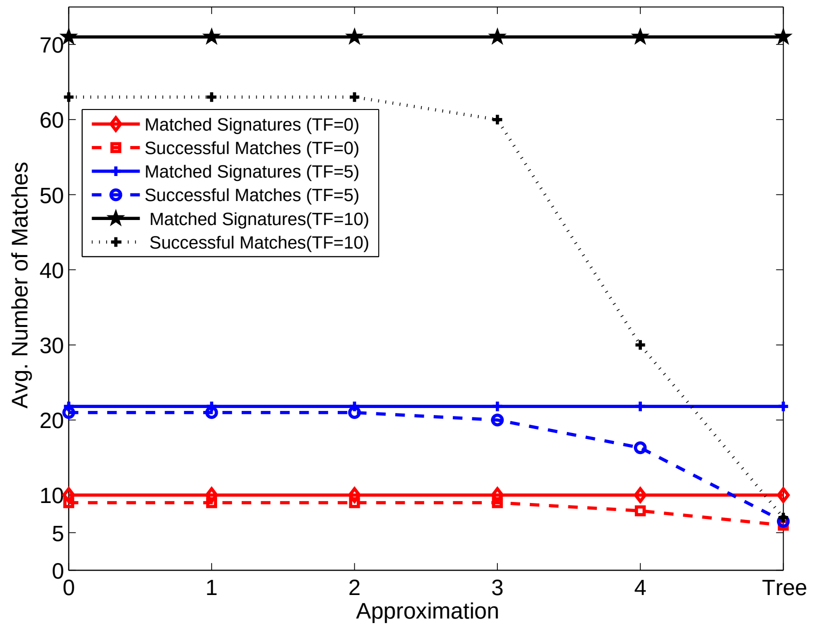


(b) Proposed Method

Figure 4.5: Comparison of results for the ORL face database using Harris

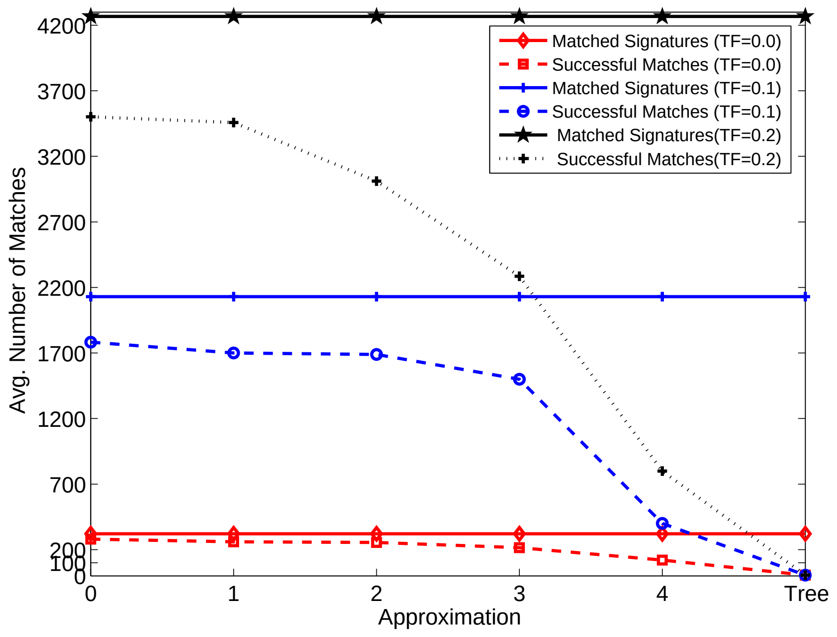


(a) Existing Method

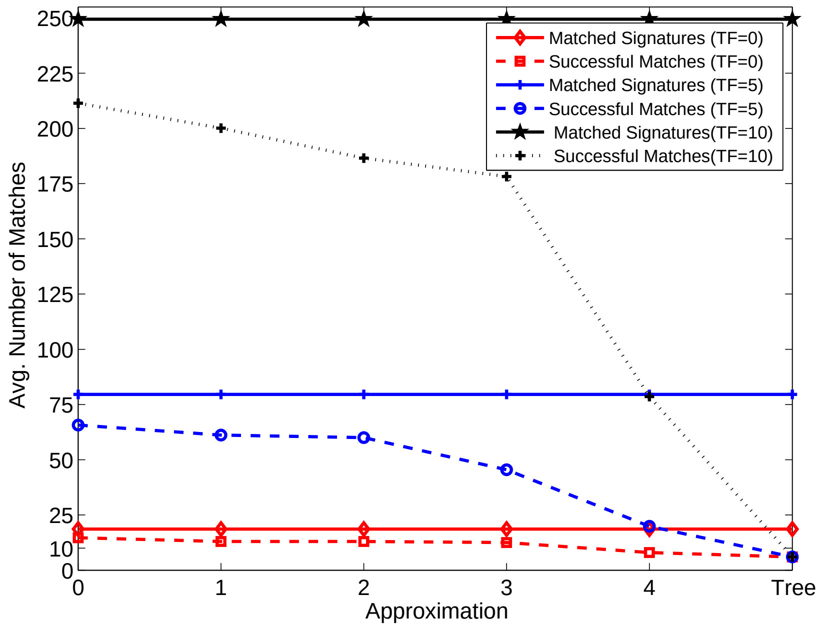


(b) Proposed Method

Figure 4.6: Comparison of results for the ORL face database using Wedge

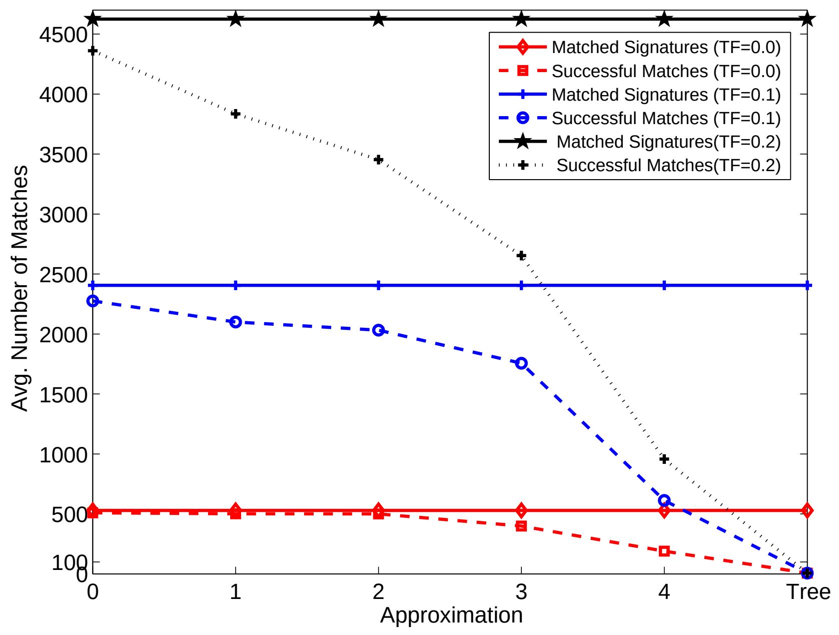


(a) Existing Method

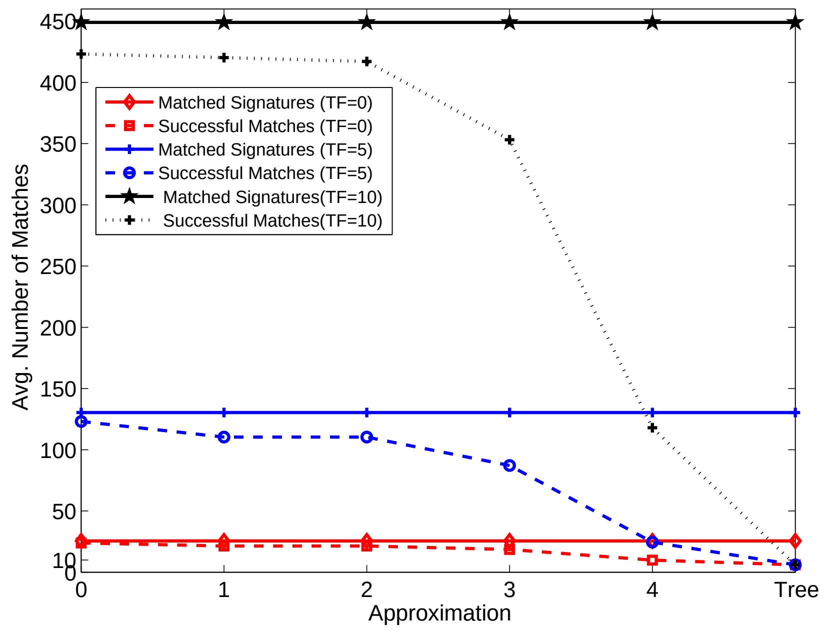


(b) Proposed Method

Figure 4.7: Comparison of results for the binary shape database using SUSAN

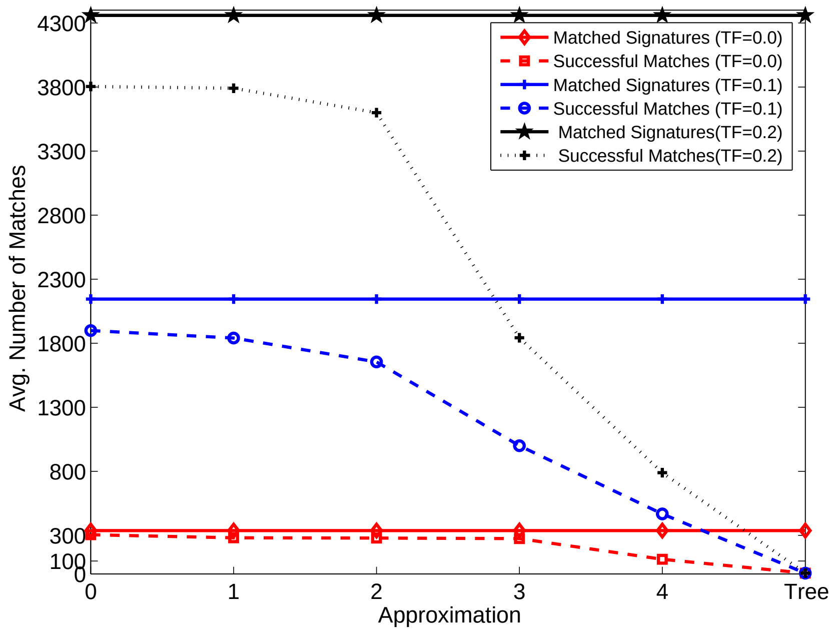


(a) Existing Method

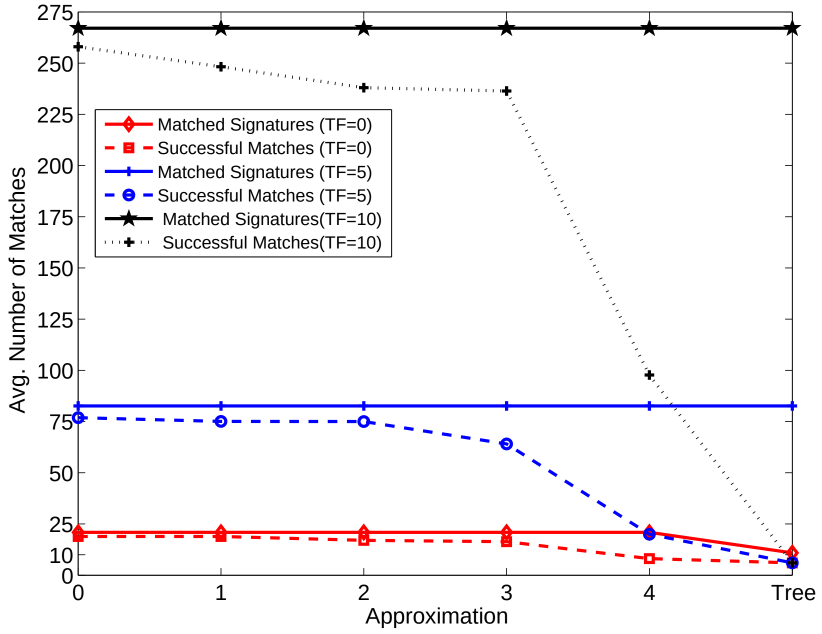


(b) Proposed Method

Figure 4.8: Comparison of results for the binary shape database using Harris

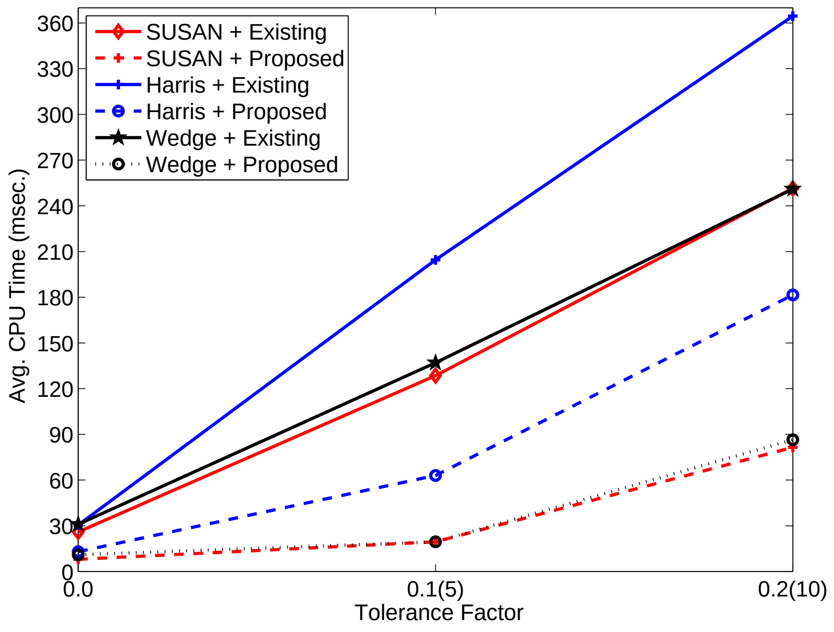


(a) Existing Method

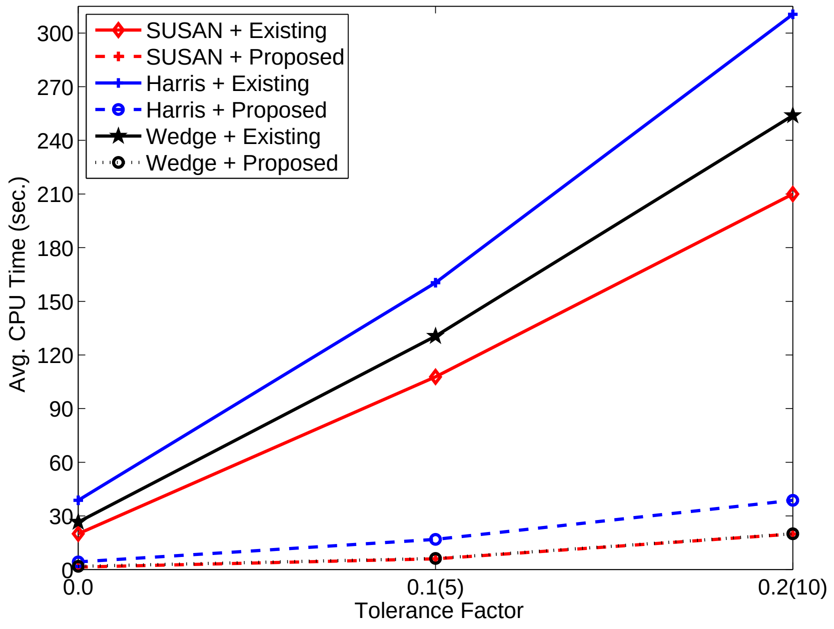


(b) Proposed Method

Figure 4.9: Comparison of results for the binary shape database using Wedge



(a) ORL face database



(b) Binary shape database

Figure 4.10: Comparison of CPU times

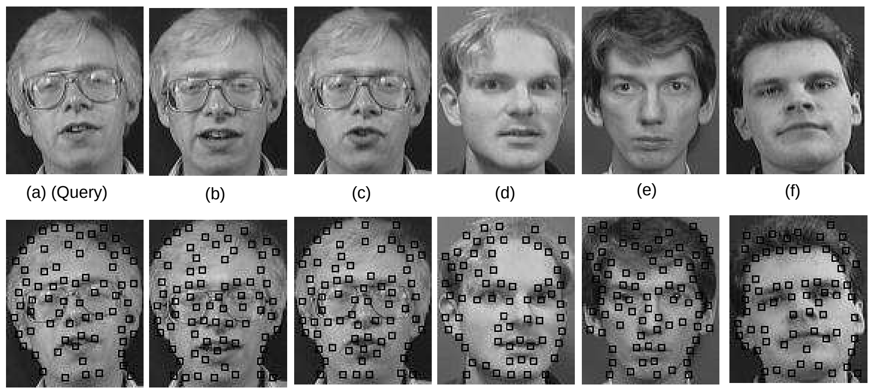


Figure 4.11: Example query and output images from the ORL face database

This is evident from Figure 4.11 where we show a query image and the retrieved images (Harris detector, $TF = 0.0$, $\gamma = 0.5$, n^{th} approximation). Here we can observe that only 3 of the 10 images for the individual in query image are retrieved by the system along with 3 apparently unexpected images of different individuals. However, if we observe these images with corner points marked more closely (using Harris detector), it may become obvious that even though the shapes in those images are different, their symbolic image representations are spatially similar.

We have also observed general behavior of the system with respect to variations in TF and approximation. For smaller approximations, lesser depths of quadtrees are compared against potential candidate images as determined by the signature matching process. When approximation is 0, only the root level is considered and as a result almost all of the signatures are successfully matched in the second stage. Also, the number of retrieved signatures increases along with an increase in TF. These results are in accordance with the general concept of the overall system.

If we compare results of the three corner detection methods, we can see that in general, Harris detector results in higher number of retrieved signatures in the first stage. As Harris

Database Name	ORL Face Database			Binary Shape Database		
Detection Method	Min. Depth	Max. Depth	Feature Points	Min. Depth	Max. Depth	Feature Points
SUSAN	7	8	36-108	4	11	5-191
Harris	6	7	43-82	5	9	10-202
Wedge	6	8	33-100	4	10	4-164

Table 4.2: Summary of characteristics of feature images

detector is based on image derivatives, it is expected to be more "consistent" across the images in a database when there is no noise. This is specially true for the ORL face database where the intensity characteristics of the images are different from each other. Since the Harris detector behaves consistently, the output symbolic images are spatially more similar to each other than those for the SUSAN or the Wedge detector. As a result we have higher number of retrieved signatures. This can be both good or bad since there is no ground rule and the ultimate choice of corner detector depends entirely on the image database and expectation of the user. Table 4.2 provides the comparison of some feature image characteristics for the three corner detection methods on the ORL face database and the binary shape database.

Figure 4.12 shows the result of changing the distance threshold value for all three tolerance factors for the spatial database. Results presented here are only for Harris detector. As mentioned before, a threshold corresponding to a measured distance of 0.0 is a special case in which an exact match between the query image and the database images is sought. On the other hand, a distance of 1.0 does not filter anything in the second phase of indexing process and will accept all of the retrieved images that have passed the signature filtering phase. We also see from Figure 4.12 that for $TF = 0$, the proposed system retrieves only 3.5 signatures on average, where there are actually 4 spatially similar images in the database against a given query image. This is because even if the images are spatially similar, $TF = 0$ is very restrictive and allows searches only for exact signature match. As a result, it does not retrieve the expected images. Indeed, the combination of different TF and γ values

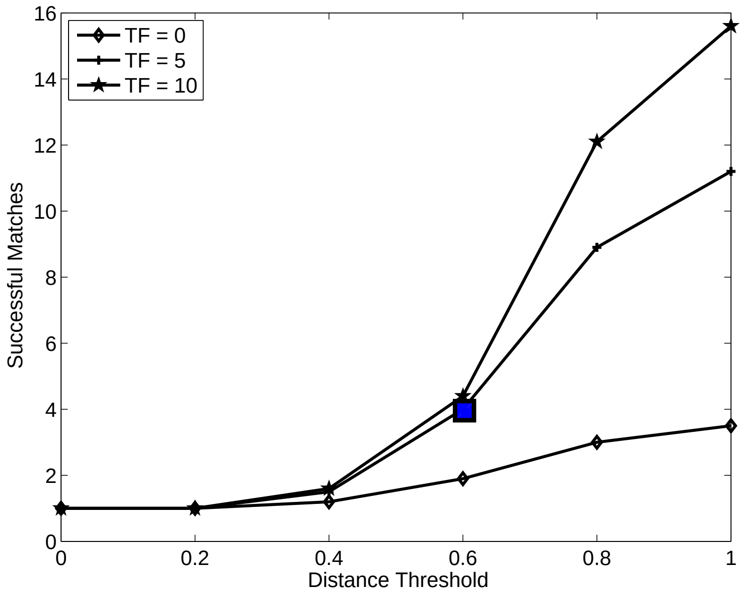


Figure 4.12: Effect of changing distance threshold on the spatial database

gives the system a wider range of customization. For this database, $TF = 5$ and $\gamma = 0.6$ seems like a reasonable choice, as it results in the expected 4 matched images for a query image (marked by a small square in Figure 4.12). For a real application where the expected output is not known, the parameters can be tuned manually or by incorporating some kind of relevance feedback from the user.

4.5 Summary

This chapter describes the primary contribution of the thesis, the new signature representation scheme. The representation method and the motivation behind the algorithm is discussed in detail. It is also formally proved that the new signature representation does not result in any new false negative. Experimental results on three different databases with varying control parameters is presented. From the results, we see that the new signature representation indeed generates fewer number of matched signatures and consequently, improved CPU time compared to the existing scheme. Comparison and analysis of the results for the three different corner detection methods, namely, SUSAN, Harris and the Wedge detector have also been presented.

Chapter 5

Conclusions and Future Works

In this thesis, we have presented a new signature representation method for a two-level spatial similarity-based image retrieval system using quadtrees. The key contributions of the thesis can be summarized as follows:

- The new signature representation scheme presented in this thesis results in significantly fewer number of matched signature in the first level of filtering compared to the existing scheme, and hence, improves the overall system performance considerably.

- It is mathematically proven that the new signature representation scheme does not carry any risk of generating new false negatives, which is a condition that every CBIR system must satisfy.

- As the system is heavily dependent on the output of the underlying corner detection method, extensive comparison of results for three different corner detection methods, namely, SUSAN, Harris and the Wedge detector is also presented.

The system presented here is highly customizable in terms of output by using combination of different values for the tolerance factor and the distance threshold. Hence, the system can be improved significantly by incorporating relevance feedback. User feedback, in turn, may allow the system to optimally auto-tune the parameters. Besides user feedback, the work may be extended to incorporate other image features such as image color and texture, so that the best match against a query can be determined.

Bibliography

[1] I. Ahmad and W. I. Grosky. Indexing and retrieval of images by spatial constraints. *Journal of Visual Communication and Image Representation*, 14(3):291–320, 2003.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2): 322–331, 1990.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] S. Berreti, A. D. Bimbo, and P. Pala. Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Transactions on Multimedia*, 2(4):225–239, 2000.

[5] S. Berreti, A. D. Bimbo, and E. Vicario. Weighted walkthroughs between extended entities for retrieval by spatial arrangement. *IEEE Transactions on Multimedia*, 5(1): 52–70, 2003.

[6] C. Chang and T. Wu. An exact match retrieval scheme based upon principal component analysis. *Pattern Recognition Letters*, 16(5):465–470, 1995.

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

[8] S. K. Chang, Q. Y. Shi, and C. W. Yan. Iconic Indexing by 2-D Strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):413–428, May 1987.

[9] Y.-I. Chang, B.-Y. Yang, and W.-H. Yeh. A generalized prime-number-based matrix strategy for efficient iconic indexing of symbolic pictures. *Pattern Recognition Letters*, 22(6-7):657–666, 2001.

[10] Y.-I. Chang, B.-Y. Yang, and W.-H. Yeh. A bit pattern based matrix strategy for efficient iconic indexing of symbolic pictures. *Pattern Recognition Letters*, 24(1-3): 537–545, 2003.

[11] Y.-I. Chang, H.-Y. Ann, and W.-H. Yeh. A unique-ID-based matrix strategy for efficient iconic indexing of symbolic pictures. *Pattern Recognition*, 33:1263–1276, 2000.

[12] X. Chen and I. S. Ahmad. Shape-based Image Retrieval using k-means Clustering and Neural Networks. In *Proceedings of the IEEE Pacific-Rim Symposium on Image and Video Technology*, pages 893–904, Santiago, Chile, 2007.

[13] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[14] S. P. Dandamudi and P. G. Sorenson. An empirical performance comparison of some

variations of the k-d tree and BD tree. *International Journal of Computing and Information Sciences*, 14(3):134–158, 1985.

[15] R. Datta, D. Joshi, and J. Wang. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Computing Surveys*, 40(2):5:1–5:60, April 2008.

[16] M. N. Do and M. Vetterli. Wavelet-Based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Transactions on Image Processing*, 11 (2):146–158, 2002.

[17] C. Faloutsos and S. Christodoulakis. Signature files: an access method for documents and its analytical performance evaluation. *ACM Transactions on Information Systems*, 2(4):267–288, 1984.

[18] C. Faloutsos and K.-I. Lin. Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 163–174, New York, NY, USA, 1995. ACM.

[19] M. Flickner, H. S. , W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC System. *Computer*, 28(9):23–32, 1995.

[20] V. Gudivada. $\Theta\Re$-String: a geometry-based representation for efficient and effective retrieval of images by spatial similarity. *IEEE Transactions on Knowledge and Data Engineering*, 10:504–512, 1998.

[21] V. N. Gudivada and V. V. Raghavan. Content-based image retrieval systems-guest editor's introduction. *Computer*, 28(9):18–22, 1995.

[22] D. S. Guru and P. Punitha. An invariant scheme for exact match retrieval of symbolic images based upon principal component analysis. *Pattern Recognition Letters*, 25(1): 73–86, 2004.

[23] A. Guttman. R-trees: a dynamic index structure for spatial searchings. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 47–57, New York, NY, USA, 1984. ACM.

[24] G. M. Haley and B. S. Manjunath. Rotation invariant texture classification using a complete space-frequency model. *IEEE Transactions on Image Processing*, 8(2): 255–269, May 1999.

[25] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[26] C. Hashizume. Focused color intersection with efficient searching for object detection and image retrieval. In *Proceedings of the International Conference on Multimedia Computing and Systems*, page 0229, Washington, DC, USA, 1996. IEEE Computer Society.

[27] D. C. Hoaglin, F. Mosteller, and J. W. Tukey. *Understanding Robust and Exploratory Data Analysis*. Wiley, New York, 1983.

[28] F.-J. Hsu, S.-Y. Lee, and B.-S. Lin. 2D C-Tree Spatial Representation for Iconic Image. *Journal of Visual Languages and Computing*, 10(2):147 – 164, 1999.

[29] P. W. Huang and Y. R. Jean. Using 2D C+-strings as spatial knowledge representation for image database systems. *Pattern Recognition*, 27(9):1249–1257, September 1994.

[30] E. Jungert. Extended symbolic projections as a knowledge structure for spatial reasoning. In *Proceedings of the 4th International Conference on Pattern Recognition*, pages 343–351, London, UK, 1988. Springer-Verlag.

[31] E. Jungert and S. K. Chang. An algebra for symbolic image manipulation and transformation,. In *Proceedings of the IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems*, pages 301–307, 1989.

[32] N. M. Khan and I. S. Ahmad. A New Signature for Quadtree based Image Matching. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, pages 20–27, Kuala Lumpur, Malaysia, 2009.

[33] B. Kimia. A large binary image database. Brown University, The Laboratory for Engineering Man/Machine Systems. URL: http://www.lems.brown.edu/˜dmc, 2010.

[34] A. Kumar. G-Tree: A New Data Structure for Organizing Multidimensional Data. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):341–347, 1994.

[35] L. J. Latecki and R. Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (10):1185–1190, 2000.

[36] S. Lee, M. Yang, and J. Chen. Signature file as a spatial filter for iconic image database. *Journal of Visual Languages and Computing*, 3(4):373–397, 1992.

[37] S.-Y. Lee and F.-J. Hsu. 2d c-string: a new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1087, 1990.

[38] J. Li, R. M. Gray, and R. A. Olshen. Multiresolution image classification by hierarchical modeling with two dimensional hidden Markov models. *IEEE Transactions on Information Theory*, 46(5):1826–1841, 2000.

[39] B. S. Manjunath, J. Ohm, V. V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6): 703–715, 2001.

[40] M. D. Mariscoi, L. Cinque, and L. Levialdi. Indexing pictorial documents by their content: a survey of current techniques. *Image and Vision Computing*, 15(2):119–141, 1997.

[41] H. P. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, Stanford University, Stanford, CA, USA, 1980.

[42] A. Natsev, R. Rastogi, and K. Shim. WALRUS: a similarity retrieval algorithm for image databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(3): 301–316, 2004.

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

[44] D. Papadias, T. Sellis, Y. Theodoridis, and M. J. Egenhofer. Topological relations in the world of minimum bounding rectangles: a study with r-trees. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 92–103, New York, NY, USA, 1995. ACM.

[45] G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora. Virtual Images for Similarity Retrieval in Image Databases. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):951–967, 2001.

[46] E. G. Petrakis, C. Faloutsos, and K.-I. D. Lin. Imagemap: An image indexing method based on spatial similarity. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):979–987, 2002.

[47] G. Petrakis, A. Diplaros, and E. Milios. Matching and Retrieval of Distorted and Occluded Shapes Using Dynamic Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1501–1516, 2002.

[48] J. L. Pfaltz, W. J. Berman, and E. M. Cagley. Partial-match retrieval using indexed descriptor files. *Communications of the ACM*, 23(9):522–528, 1980.

[49] W. K. Pratt. *Digital Image Processing: PIKS Inside*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[50] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[51] P. Punitha and D. S. Guru. Symbolic image indexing and retrieval by spatial similarity: An approach based on B-tree. *Pattern Recognition*, 41(6):2068–2085, 2008.

[52] J. T. Robinson. The K-D-B-tree: a search structure for large multidimensional dynamic indexes. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 10–18, New York, NY, USA, 1981. ACM.

[53] J. Rogers. *The Dictionary of Clichs*. Ballantine Books, New York, 1985.

[54] J. Rothnie, B. James, and T. Lozano. Attribute based file organization in a paged memory environment. *Communications of the ACM*, 17(2):63–69, 1974.

[55] H. Samet. Distance transform for images represented by quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(3):298–303, 1982.

[56] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[57] J. R. Smith and S. Chang. VisualSEEk: a fully automated content-based image query system. In *Proceedings of the 4th ACM international conference on Multimedia*, pages 87–98, New York, NY, USA, 1996. ACM.

[58] S. M. Smith and J. M. Brady. SUSAN—A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[59] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177 – 280, 2007.

[60] E. Vincent and R. Laganire. Detecting and matching feature points. *Journal of Visual Communication and Image Representation*, 16(1):38–54, 2005.

[61] J. Z. Wang, N. Boujemaa, A. D. Bimbo, D. Geman, A. G. Hauptmann, and J. Tesić. Diversity in multimedia information retrieval research. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 5–12, New York, NY, USA, 2006. ACM.

[62] Y. Wang. Image indexing and similarity retrieval based on spatial relationship model. *Information Sciences*, 154(1-2):39–58, 2003.

[63] N. Xing and I. S. Ahmad. Shape-based Image Retrieval. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, pages 543–547, Kuala Lumpur, Malaysia, 2009.

[64] H. Yu and J. Yang. A direct LDA algorithm for high-dimensional data with application to face recognition. *Pattern Recognition*, 34:2067–2070, 2001.

[65] D. Zhang and G. Lu. Shape-based image retrieval using generic fourier descriptor. *Signal Processing: Image Communication*, 17(10):825 – 848, 2002.

Vita Auctoris

Naimul Mefraz Khan was born in 1984 in Dhaka, Bangladesh. He received his Bachelors degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2008. His research interests include image processing, image retrieval, computer vision, machine learning and pattern recognition.