A Two-Stage Framework for Polygon Retrieval

By

Tung Lun Hsing

Supervised By

Prof. Irwin King

A Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Philosophy

Division of Computer Science & Engineering

The Chinese University of Hong Kong

June 1997

Abstract

Query-by-shape is a fundamental operation in image database systems as it provides an intuitive way to access objects by their outlines. Shape matching and retrieval are the keys of shape query processing.

We propose a two-stage framework for polygon retrieval which incorporates both qualitative and quantitative measures of polygons in the first and second stage respectively. The first stage uses Binary Shape Descriptor (BSD) as a mean to prune the search space. The second stage uses any available polygon matching or similarity measuring technique to compare model polygons with the target polygon. This two-stage framework uses a combination of model-driven approach and data-driven approach. It is more efficient than model-driven approach since it reduces the number of polygons needed to be compared. By using binary string as index, it also avoids the difficulty and inefficiency of maintaining complex multi-dimensional index structure. This two-stage framework can be incorporated into image database systems for providing query-by-shape facility.

We also propose two similarity measures for polygons, namely Multi-Resolution Area Matching (MRAM) and Minimum Circular Error Bound (MCEB), which can be used in the second stage of the two-stage framework. The MRAM method is an area-based technique incorporating multi-resolution Quadtree area coding. The MCEB method is developed based on an intuitive human perception of polygon resemblance by measuring the distance between corresponding vertices of different polygons.

Experiments show that the two-stage approach is more efficient than the Normalized Coordinate System (NCS) approach with KD-Tree indexing. We also compare the efficiency of the proposed MRAM method, the MCEB method, with the NCS method and the Hausdorff Distance method. Experiments show that the MRAM method is more efficient than the other three methods in terms of running time. Visual ranking of polygons produced by the aforesaid methods are also compared and experiments show that the MCEB method produces better ranking than the other three methods.

Acknowledgement

I would like to begin by thanking my parents, my sisters, and my grandmother, for the support they have given me throughout the years. I would also like to dedicate this work to my beloved grandfather.

My research advisor, Prof. Irwin King, has been my primary professional and intellectual influence during my time here at the Chinese University of Hong Kong. I am grateful to him not only for his ideas, but also for the way of research he taught me.

I would like to thank my closest friends here at the Chinese University of Hong Kong. I am especially grateful to Wong Tai Man, Law Kwok Wai and Kuok Chan Man for the care they shown me, the support they gave me, and all the fun that we shared.

Finally, I would also like to acknowledge my friends outside of the Chinese University of Hong Kong. I would like to thank Sonny Lau, Cecilia Leung, Li Ngai, Philip Ng, and Ronald Wong.

Contents

Abstract **i**

Acknowledgement **ii**

1 Introduction **1**

2 Literature Survey **8**

2.1 The Freeman Chain Code Approach 8

2.2 The Moment Approach 10

2.3 The Rectangular Cover Approach 12

2.4 The Potential-Based Approach 15

2.5 The Normalized Coordinate System Approach 17

2.6 The Hausdorff Distance Method 20

2.7 The PCA Approach 22

3 Binary Shape Descriptor **26**

3.1 Basic idea 26

3.2 Standardized Binary String Descriptor 27

3.3 Number of equivalent classes for n-gons 28

4 The Two-Stage Framework **30**

5 Multi-Resolution Area Matching **33**

5.1 The idea 33

5.2 Computing MRAI . . . 34
5.3 Measuring similarity using MRAI . . . 36
5.4 Query processing using MRAM . . . 38
5.5 Characteristics and Discussion . . . 40

6 Circular Error Bound and Minimum Circular Error Bound **41**
6.1 Polygon Matching using Circular Error Bound . . . 41
6.1.1 Translation . . . 43
6.1.2 Translation and uniform scaling in x-axis and y-axis directions . . . 45
6.1.3 Translation and independent scaling in x-axis and y-axis directions 47
6.2 Minimum Circular Error Bound . . . 48
6.3 Characteristics . . . 49

7 Experimental Results **50**
7.1 Setup . . . 50
7.1.1 Polygon generation . . . 51
7.1.2 Database construction . . . 52
7.1.3 Query processing . . . 54
7.2 Running time comparison . . . 55
7.2.1 Experiment I . . . 55
7.2.2 Experiment II . . . 58
7.2.3 Experiment III . . . 60
7.3 Visual ranking comparison . . . 61
7.3.1 Experiment I . . . 61
7.3.2 Experiment II . . . 62
7.3.3 Experiment III . . . 63
7.3.4 Conclusion on visual ranking experiments . . . 66

8 Discussion **68**
8.1 N-ary Shape Descriptor . . . 68

8.2 Distribution of polygon equivalent classes 69
8.3 Comparing polygons with different number of vertices 72
8.4 Relaxation of assumptions 73
8.4.1 Non-degenerate 74
8.4.2 Simple 74
8.4.3 Closed 76

9 Conclusion **78**

Bibliography **80**

List of Tables

2.1 Summary of the characteristics of different methods 25

3.1 N-gons and number of their equivalent classes 29

5.1 3 MRAIs at resolution level 1 37
5.2 MRAM similarity measures with different p values 38

7.1 Polygon databases for running time experiment I 56
7.2 Average running time of experiment I 56
7.3 Average running time of experiment II 58
7.4 Normalized average running time of experiment II 59
7.5 Average running time of experiment III 61

8.1 SBSD, SNSD^4, and SNSD^8 of triangles 70
8.2 Distribution of equivalent classes of 4-sided polygons 70
8.3 Distribution of equivalent classes of 5-sided polygons 71
8.4 Distribution of equivalent classes of 6-sided polygons 71
8.5 Distribution of equivalent classes of 7-sided polygons 71
8.6 Restrictions for different techniques 77

List of Figures

1.1 Query-by-shape 2
1.2 Open, complex, and degenerate polygons 4

2.1 Freeman Chain Code with 4 directions 9
2.2 Derivative of Freeman Chain Code 10
2.3 Computing the rectangular cover 13
2.4 Different rectangular covers of the same shape 15
2.5 Basic matching iteration of Potential-Based Approach 16
2.6 Problem of the Potential-Based approach 17
2.7 An example of 3D-Tree 18
2.8 Normalizing coordinate list 19
2.9 Problem of Hausdorff Distance method 22
2.10 Attributed graph representation of a polygon 23
2.11 Undesirable visual ranking produced by PCA approach 24

3.1 Polygons and their BSDs 27
3.2 Different BSDs for the same polygon 27
3.3 Polygon in an equivalent class 29

4.1 An example of polygon selection at the first stage of the two-stage framework 31

5.1 Computing Multi-Resolution Area Information (MRAI) 35

6.1 An intuitive definition of similar polygons 42

6.2 Intersection of Circular Error Bounds 44
6.3 $\mathbb{S}_{ij}$ and its intersection 46
6.4 $\mathbb{E}_{ij}$ and its intersection 47

7.1 Orientation standardization and scale normalization of polygons 53
7.2 Average running time of experiment I 57
7.3 Average running time of experiment II 59
7.4 Polygon data for visual ranking experiment I 62
7.5 Polygon data for visual ranking experiment II 63
7.6 Visual ranking produced by the Hausdorff Distance method in visual ranking experiment II 64
7.7 Polygon data for visual ranking experiment III 64
7.8 Visual ranking produced by the MRAM method in visual experiment III . 65
7.9 Visual ranking produced by the NCS method in visual experiment III . . . 65
7.10 Similar query result I 66
7.11 Similar query result II 67

8.1 Triangles with different interior angles 69
8.2 Polygon simplification using curvature information 72

Chapter 1

Introduction

Shape matching and shape similarity measuring are challenging problems in image processing and computational geometry. Shape matching problem concerns with determining whether two shapes match each other or not under user specified tolerance. Shape similarity measuring problem concerns with finding out numerical measures on how two shapes are similar to each other. Both problems involve the following issues:

1. **Shape representation and computation**

   This issue is concerning how a shape can be represented in terms of its shape properties and whether the representation is invariant to translation, scale, and orientation. Besides, the method to extract the representation from a shape and the computation involved are also concerned. In general, translation invariant, scale invariant, and orientation invariant are desirable characteristics of shape representations but not necessarily. For example, user may want to retrieve a shape in particular orientation. An orientation invariant representation will not be suitable in such situation.

2. **Similarity measure**

   Given a representation scheme, we then have to come up with matching or similarity measuring methods that work on the representation. The goal is that such comparison methods should be efficiently computable and should produce visual ranking of shapes resembling human perception.

3. **Retrieval method**

   Given a representation scheme and its comparison methods, we then have to decide how to organize and store the data. The goal is to enable efficient search for, or access to, shapes in the database. We have to determine whether index structure can be used on the representation scheme, and select suitable index structure if index can ever be used.

Shape matching and shape similarity measuring have many applications. For example, they can be applied in hand writing recognition systems and image database systems. Query-by-shape is a fundamental operation in image database systems. It provides an intuitive way to access objects by their outlines. The task of a shape query is to find out the set of shapes, out of a set of model shapes, that are similar to or matched a given target shape. Figure 1.1 shows how an image database system provides the query-by-shape facility.

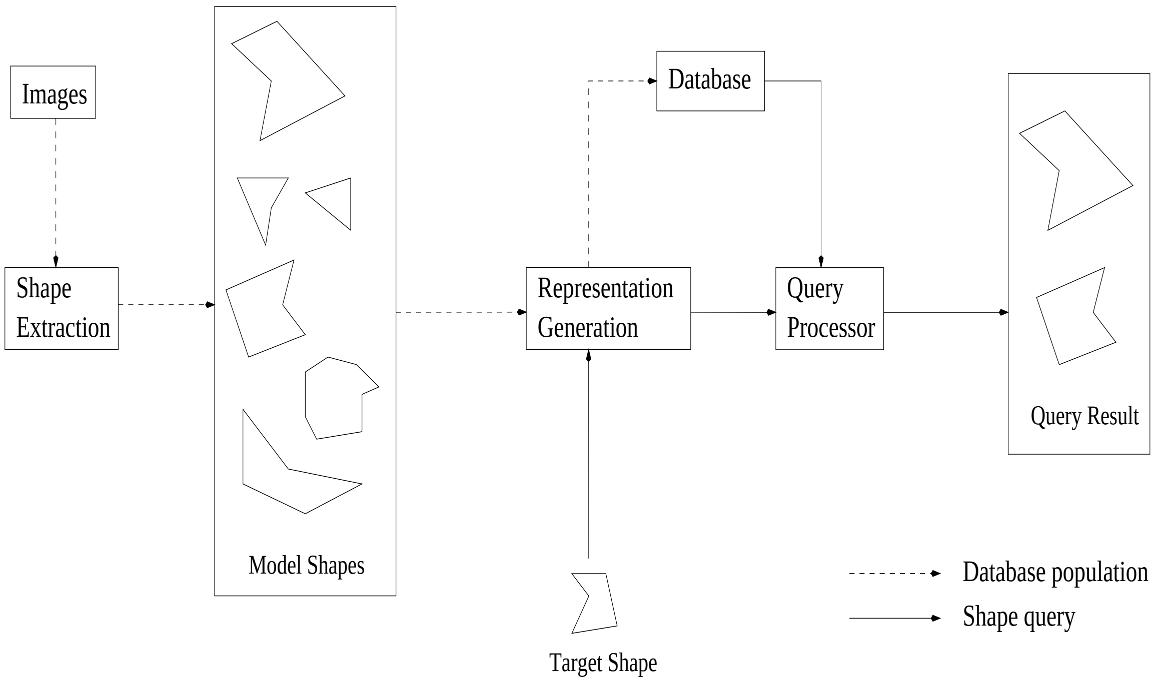


Figure 1.1: Query-by-shape

Assumptions

In this paper, we only concentrate on the issues of shape representation, shape similarity measure, and shape indexing. Extracting shapes from images are not our concern. We assume that when an image is added into an image database, it is associated with a set of shapes representing the objects inside the image. This can be an automated process by incorporating any available feature extraction techniques like [45, 22, 21, 18], or it may require user intervention. Therefore, we assume that the inputs to our work are the shapes extracted from images and the shapes are in the form that we described below.

Our work concentrates on 2D polygonal shapes instead of arbitrary shapes. We will use the word *shape* and the word *polygon* interchangeably hereafter. At present stage, our work only handles simple non-degenerate closed polygons as defined below. The relaxation of these constraints will be discussed in Chapter 8.

Definition 1 A polygon is represented by an ordered list of vertices $P = \{V_1, V_2, \cdots, V_n\}$, where n is the number of vertices of the polygon and $V_i \in \mathbb{R}^2$.

Definition 2 A polygon is simple if no two edges of the polygon cross each other.

Definition 3 A polygon is non-degenerate if $\nexists 1 \leq i \leq n$ such that V_i, $V_{(i+1) \bmod n}$, and $V_{(i+2) \bmod n}$ are collinear.

Figure 1.2 shows several examples of polygons that do not adhere to the above definitions.

Shape queries

We consider two kinds of shape queries, namely *matching queries* and *similarity queries*. A matching query tries to retrieve all model shapes in a database which are matched with the target shape subject to some predefined tolerances. A similar query tries to retrieve a number of model shapes that are most similar to the target shape. The tasks of these two kinds of queries are defined as follows:

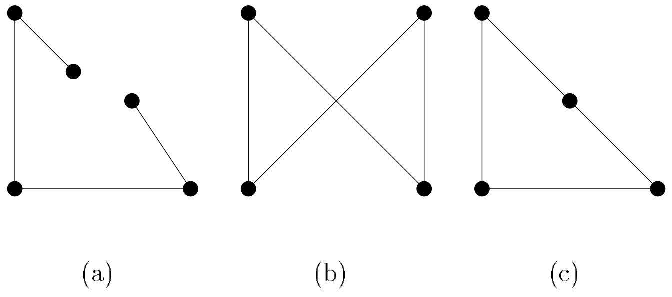


Figure 1.2: Open, complex, and degenerate polygons

(a) An open polygon. (b) A polygon with crossing edges. (c) A degenerate polygon.

1. **Matching query**

   $\boldsymbol{R} = \{P_i \mid P_i \in \boldsymbol{P} \ \wedge \ MATCH(P_i, T)\}$ where $\boldsymbol{R}$ is the result of the query, $\boldsymbol{P}$ is the set of model polygons, T is the target polygon, and $MATCH(\cdot)$ denotes a polygon matching technique.

2. **Similarity query**

   $\boldsymbol{R} = \{P_i \mid P_i \in \boldsymbol{P} \ \wedge \ 1 \leq i \leq n \ \wedge \ P_1 \leq P_2 \leq \cdots \leq P_m\}$ where $\boldsymbol{R}$ is the result of the query, $\boldsymbol{P}$ is the set of model polygons, n is the number of polygons to be included in $\boldsymbol{R}$, m is the number of polygons in $\boldsymbol{P}$, $n \leq m$, and $P_1 \leq P_2 \leq \cdots \leq P_m$ is the ranking produced by a polygon similarity measuring technique based on the degree of similarity between the model polygons and the target polygon.

Most polygon similarity measuring techniques generate numerical values to measure the similarity between polygons. Such techniques can be used to handle both kind of queries. Matching queries can be handled by thresholding the numerical values. Similar queries can be handled by sorting polygons with respect to the numerical values.

Contributions

The contributions of our work include the proposal of a two-stage framework for polygon retrieval, a polygon similarity measuring technique named Multi-Resolution Area Matching (MRAM), a polygon matching technique using Circular Error Bound (CEB), and a polygon similarity measure called Minimum Circular Error Bound (MCEB). We have also studied the characteristics of several existing methods and compare their performance with our methods.

The two-stage framework for polygon retrieval

Considerable work has been carried out on shape matching problem. A literature survey is presented in Chapter 2. Traditional techniques use model-driven approach in which the target shape is compared individually against each model shape [14, 15, 9]. This approach is inefficient because of its linear searching complexity. Other techniques use data-driven approach in which features of shapes are extracted and mapped into a multi-dimensional index structure. Matching is conducted by performing searching in the index tree [23, 28, 29]. The efficiency of data-driven approach highly depends on the efficiency of the Point Access Method (PAM, a data structure that support storage and retrieval of points in a multi-dimensional space) used. Roughly speaking, a large number of dimension is required when mapping shape features into multi-dimensional index structures and PAMs are inefficient under this situation. Moreover, when using PAM, additional computation is required to maintain the complex multi-dimensional index structure whenever the database is modified.

We propose a two-stage framework for the polygon matching and retrieval task. This two-stage frame uses a combination of model-driven approach and data-driven approach:

1. The first stage of this two-stage framework incorporates qualitative measure of polygons as a filtering function. It maps polygons into binary string using the Binary Shape Descriptor (BSD) [6] technique and uses these binary string as an index.

This index is then used retrieval subset of model polygons from the database for the second stage matching.

2. The second stage of the framework incorporates any available quantitative polygon matching or similarity measuring technique to perform matching on the subset of polygons produced in the first stage. Techniques using both model-driven approach and data-driven approach can be used.

This two-stage framework is more efficient than model-driven approach since it reduces the number of polygons needed to be compared. It also avoids the difficulty and inefficiency of manipulating complex multi-dimensional index structures. Instead, it uses string as index which is well studied and efficient indexing techniques are available. Even data-driven approach technique is used in the second stage, the framework helps to reduce the size of the index structure and thus improve the efficiency of the technique.

Multi-Resolution Area Matching

The MRAM technique we proposed is an area-based polygon similarity measuring technique incorporating Quadtree [37] area coding. Its multi-resolution nature makes it possible to further speed up the query processing task under the two-stage framework.

Polygon matching using Circular Error Bound

We propose a polygon matching technique using CEB which is based on an intuitive human concept of polygon resemblance. This technique can only be used to determine whether two polygons resemble each other under certain transformations and user specified tolerance.

Minimum Circular Error Bound

We extend the idea of CEB method and propose a polygon similarity measuring technique name MCEB. The MCEB technique computes translation invariant numerical values from the distance between corresponding vertices of different polygons as the similarity measures between these polygons.

Characteristics study and performance evaluation

We have studied the characteristics of the Freeman Chain Code method [14, 15], the Moment method [25], the Rectangular Cover method [23], the Potential-based method [9], the Normalized Coordinate System method [28, 29], the Hausdorff Distance method [20, 2], and the PCA method [44]. We also compare the performance of the Normalized Coordinate System method and the Hausdorff Distance method with our MRAM method and MCEB method.

Organization of this thesis

The rest of this thesis is organized as follows. A literature survey is presented in Chapter 2. Before we describe the two-stage framework in details in Chapter 4, we introduce the idea of Binary Shape Descriptor (BSD) and the computation of Standardized Binary Shape Descriptor (SBSD) in Chapter 3 since they play an important role in the two-stage framework. We propose the MRAM technique in Chapter 5. Polygon matching technique using CEB, as well as the translation invariant polygon similarity measure MCEB, are described in Chapter 6 We present and discuss the experimental result in Chapter 7. Discussion and possible extension to the two-stage framework are presented in Chapter 8. Finally, conclusion is made in Chapter 9.

Chapter 2

Literature Survey

Considerable work has been carried out on shape matching and shape similarity measuring problem. In this section, we will discuss some of these methods such as the Freeman Chain Code approach, the moment approach, the rectangular cover approach, the potential-based approach, the normalized coordinate system method, the Hausdorff Distance method, and the PCA method. Table 2.1 gives a summary of the characteristics of these aforesaid methods.

2.1 The Freeman Chain Code Approach

Idea

The Freeman Chain Code method is proposed in [14, 15] and has been used in several works such as [38, 34]. In this method, a shape is first scan-converted into a frame buffer. The Freeman Chain Code of the digitized shape is generated by recording the turning directions of its boundary pixels. The turning direction of the boundary pixels are recorded in 4 discrete levels so a Freeman Chain Code is a string of an alphabet consisted of 4 symbols. Figure 2.1 illustrates the computation of Freeman Chain Code. The shape matching task is carried out by comparing the Freeman Chain Codes of the given shapes. Matching of shapes is performed by substring matching of the Freeman Chain Code. The similarity between shapes can be measured by number of coincide symbols in the Chain Codes.

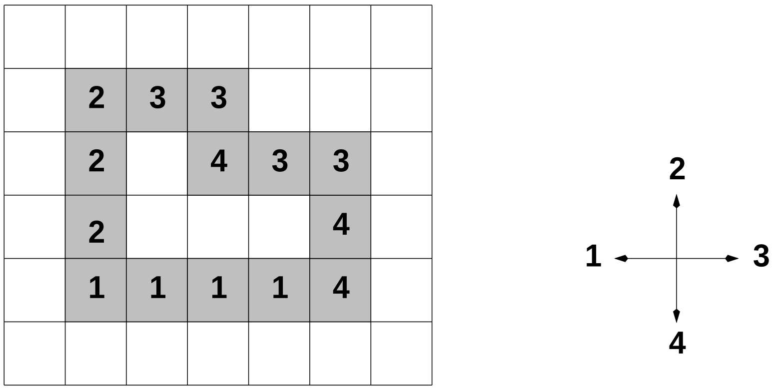


Figure 2.1: Freeman Chain Code with 4 directions

The Freeman Chain Code recorded starting at the lower left corner is 12223343344111.

Characteristics

This method is translation invariant. However, it is scale and orientation sensitive so normalization is required before the original shapes are digitized. Moreover, this method is sensitive to the selection of *anchor pixel* (the boundary pixel at where the recording of Freeman Chain Code is started). The storage complexity of this method is $O(k)$ where k is the number of boundary pixels of the digitized shape. The computational complexity is also $O(k)$. This method is indexable since the representations are simply strings.

Variations

In [7], an variance of the Freeman Chain Code method is proposed which use transformation invariant derivative of Freeman Chain Code to represent a shape. The derivative of a Freeman Chain Code is obtained by clockwise recording each convex corner, straight corner, and concave corner of the boundary pixels. Figure 2.2 illustrates the computation of derivative of Freeman Chain Code. The shapes being processing are first re-oriented by aligning their major axes with the x-axis. These normalized shapes are then scan-converted into frame buffer with different resolutions and the derivatives of Freeman Chain Code under each resolution are recorded. In coarse resolutions, shapes tend to have the same derivative. As the resolution increased, the derivatives of the shapes become distinguishable. Based on this idea, the similarity measure of two shapes is defined as the finest resolution at which the derivatives of the two shapes still remain the same.

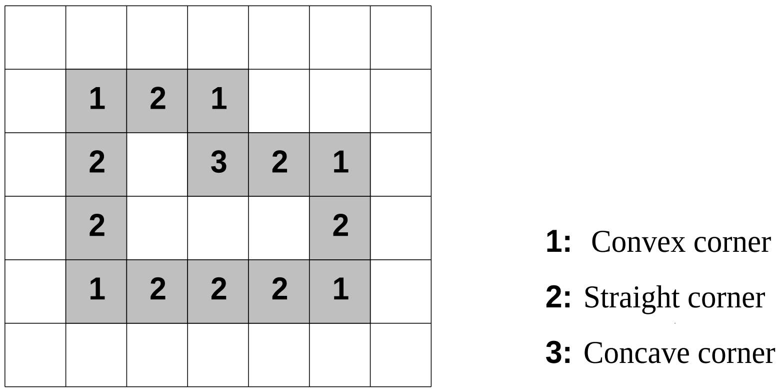


Figure 2.2: Derivative of Freeman Chain Code

The derivative recorded starting at the lower left corner is 12212132121222.

This method is translation, scale and orientation invariant since normalization is taken into account. The storage and computational complexity of this method depends on the number of resolution levels where the derivatives of Freeman Chain Code are recorded. Reasonable matching and similarity measuring results will require derivatives being recorded up to certain fine resolution or else shapes will only be able to be classified into a few number of classes.

2.2 The Moment Approach

Idea

This approach represents a shape using shape moments. Regarding the contour of a shape as a continuous 2D function, the (p, q)th moment of the function is defined as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \tag{2.1}$$

For a digitized shape which is scan-converted into a W pixels by H pixels frame buffer, its (p, q)th moment can be defined as:

$$m_{pq} = \sum_{i=1}^{W} \sum_{j=1}^{H} i^p j^q v(i, j) \tag{2.2}$$

where $v(i, j) \in \{0, 1\}$, which a value of 1 indicates that the pixel at position (i, j) is part of the digitized shape. The difference between the (p, q)th moments of two given shapes is used as the similarity measure of the two shapes. Two shapes are said to be matched with each other if the difference between their (p, q) moments is less than or equal to a predefined tolerance. More than one moments can be used together for the matching process.

Characteristics

The moment approach is translation, scale and rotation sensitive. Therefore, shapes have to go through preprocessing like translation normalization, scaling normalization and orientation normalization before they can be handled by the moment approach. For each shape, only the computed moment will be stored thus the storage requirement is $O(1)$ and the computational complexity of the shape moment is $O(WH)$. Since the shape moment is just a numerical value, it can be easily indexed for efficient retrieval.

Variations

The computational requirement of shape moment is quite large because every pixel of the frame buffer is involved in the computation. In [25], an extension is proposed to the moment approach which computes the (p, q)th moment of a shape using its boundary pixels only. Since the number of boundary pixels of a shape must be less than or equal to the number of all its pixels, the new approach is more efficient than the original one in terms of computation. In general, the number of boundary pixels of a shape is far less than the number of all its pixels and this new approach introduces a significant improvement over the original approach. From the experiment stated in [25], this new method only requires about 3% to 4% running time of the old method.

2.3 The Rectangular Cover Approach

Idea

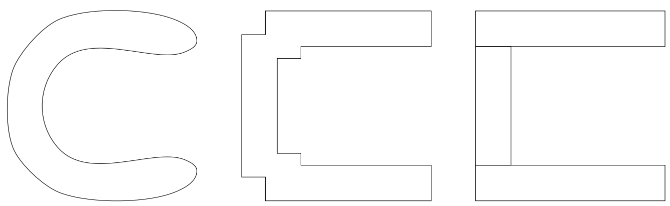
In [23], shapes are represented by rectangular covers and similarity between rectangular covers are used to measure the similarity between their corresponding shapes. A shape is first approximated by a rectilinear shape with certain resolution. Then the rectangle cover of this rectilinear shape is computed (Figure 2.3). A rectangle cover of a rectilinear shape is a set of rectangles which reconstruct the rectilinear shape when they are union together. The rectangular cover is constructed in the way that rectangles are placed sequentially with respect to the importance of the features they are representing. In other words, we can consider that the rectangles in the front of the a rectangular cover provides the basic shape features of the original shape while other rectangles are used to refine this basic shape and give details of the original shape. How well a rectangle cover represents its rectilinear shape depends on the number of rectangles allowed in the rectangle cover. A rectangle cover with few rectangles will only be able to provide a rough approximation of its rectilinear shape. A rectangle cover is stored using the following scheme:

1. For each rectangle in the rectangle cover, the coordinate of its center is stored as its position. Its width and height is also stored as its size.

2. The position of the first rectangle of a rectangle cover is used to normalize the position of other rectangles in the rectangle cover, by regarding its position as the origin.

3. The size of the first rectangle is used to normalize the size of other rectangles, by regarding both its width and height as 1.

4. The *scaling factor*, which is defined as the product of the width and the height of the first rectangle, is stored. The *distortion factor*, which is defined as the ratio of the width and the height of the first rectangle (height divided by width), is also stored.



(a) (b) (c)

Figure 2.3: Computing the rectangular cover

(a) The original shape. (b) The rectilinear approximation. (c) The rectangular cover.

To process a matching query on two given shapes, their rectangular covers are compared as follows. Rectangles in the two rectangular covers are examined in pairs, i.e. the first rectangles of the two covers are compared and the second rectangles of the two covers are compared, etc. The positions and sizes of a pair of rectangles are tested to find out whether the differences between them are within a set of predefined tolerances. The same testing procedure is applied to the scaling factors and distortion factors of the two rectangular covers as well. Only when the two rectangular covers pass all the above tests will their corresponding shapes be regarded as matched with each other.

By ignoring part of the stored information, matching of shape under specific transformation constraints can also be done. For example, to perform shape matching under scaling, the *scaling factor* of the rectangular cover is not compared in the matching process. In case that the dimension of the rectangular cover of the target shape dismatches those of model shapes, only the minimum number of rectangles are compared.

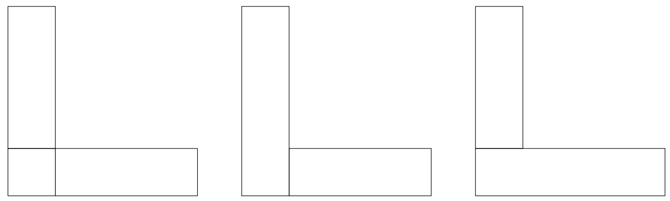
Characteristics

This approach is translation and scale invariant but rotation sensitive so shapes have to go through orientation normalization before they are handled by this approach. The storage complexity of this approach is $O(l)$ where l is the number of rectangles used in a rectangular cover. The computational complexity of comparing two rectangular cover is $O(l)$. However, computational complexity of the preprocessing depends on the method used for generating rectangular cover from a rectilinear shape. An $O(n \log n)$ algorithm is proposed in [13] for computing the rectangle cover where n is the number of vertices of the rectilinear shape. [8] proposes a parallel algorithm for this task which run on $O(m/\log m)$ Concurrent-Read-Exclusive-Write (CREW) processors in $O(\log m)$ time where m is the number of pixels in the frame buffer on where input rectilinear shapes are rendered. Index can be constructed on the rectangular covers to speed up the matching.

Discussion

One problem of this approach is that it cannot handle similar queries efficiently. The proposed way to handle similar queries is to perform matching queries in a iterative manner with tolerance relaxed in each iteration. Model shapes can then be retrieved in groups in descending order of their similarity to the target shape. However, the scheme for tolerance relaxation and the efficiency of this method is not well defined.

Another problem of this approach is that the rectangular cover of a shape is not unique. It is hard to determine which rectangular cover should be used to represent a shape if more than one rectangular covers are available. Some heuristic, e.g. choosing the rectangle cover with least number of rectangles, may help in making the decision but does not guarantee that there will not be more than one equally good representations. For example, the rectangular cover in Figure 2.4(b) and Figure 2.4(c) is better than the one in Figure 2.4(a) but they are equally good. The solution to this problem proposed in [23] is to use all equally good representations of a shape. Thus, multiple entries representing the same shape will be introduced into the system.



(a) (b) (c)

Figure 2.4: Different rectangular covers of the same shape

2.4 The Potential-Based Approach

Idea

The author of [9] introduce a shape classification technique using an artificial potential field. The proposed potential model assumes that the contour of any 2D shape is uniformly charged (artificially). The idea is to place a small shape template inside the data shape (the shape to be classified) and let the template expands subject to the repulsive force and torque it experiences in the artificial potential field set up by the contours of the template and the data shape. The expansion is terminated when the template is expanded to a size that it touches the contour of the data shape. The above procedure is applied to every template in the system. The data shape is said to be belong to the group of the template shape having the largest area after the above operation.

The Newtonian potential model is adopted in this approach so the potential related quantities can be derived analytically. This avoids the expensive numeric implementation of the discretization of the shape contours before calculating the repulsive force and torque. However, the expansion of a template inside a data shape is still an iterative process. In each iteration, the repulsive force and the torque are computed and the template shape is translated, rotated, and expanded accordingly. Figure 2.5 illustrates these steps in an iteration of the matching process. The process is stopped when the growth of area in an iteration is negligible.

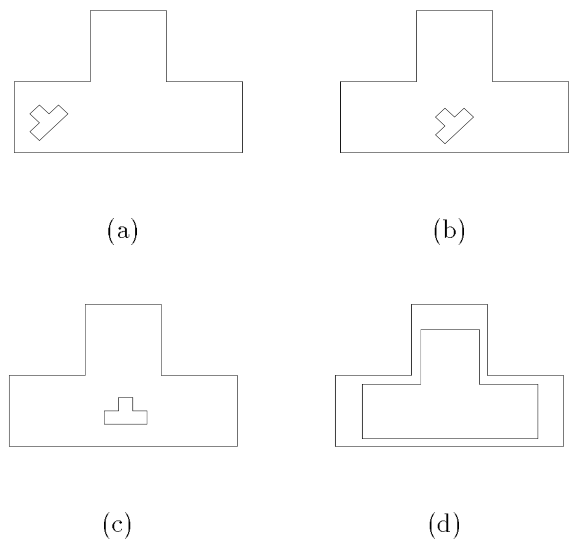


Figure 2.5: Basic matching iteration of Potential-Based Approach
(a) the template is put inside the target shape (b) translating the template (c) rotating the template (d) increasing the size of the template

Characteristics

This approach is translation, scale and orientation invariant. The storage requirement is $O(n)$ where n is the number of vertices of the templates. However, since this is an iterative approach, the computation complexity of this approach is hard to analyze and it is not stated in [9]. According to the experiments performed in [9], it takes the system about 10 seconds to classify a shape on a Sun SPARCstation 10 with a database consists of 12 templates. This method is not indexable.

Discussion

As the computational requirement of this approach is quite large as shown in the experiment in [9], this approach is not suitable for general shape matching purpose. However, it is suitable for systems which need to classify data shapes into a small number of classes. Besides, this approach has difficulty on handling concave shapes. For example, it will not work on matching the outline of the letter "H". If a small template of this shape is placed at the center of a data shape which has exactly the same "H" shape but is larger than the template, the template will only be able to expand a little bit before it touches the contour of the data shape (Figure 2.6). Thus, a more sophisticated expansion operation are needed to handle this kind of shapes.

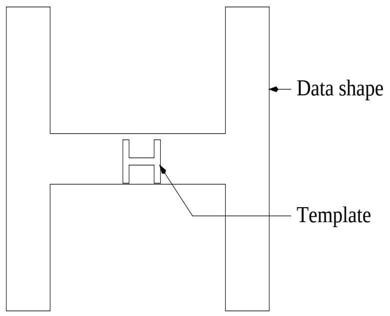


Figure 2.6: Problem of the Potential-Based approach

2.5 The Normalized Coordinate System Approach

Idea

In [28, 29], a system is proposed which uses a list of normalized coordinates to represent polygonal shapes. We will refer to this approach as Normalized Coordinate System (NCS) hereafter. These normalized coordinate lists are treated as multi-dimensional data points and are used to construct index tree using any available PAM. The KD-Tree [37] technique is adopted in [28, 29].

In principle, a KD-Tree is a binary tree with the distinction that it uses different part of a K-dimensional key at different level to construct the sub-trees, i.e. the $(i \mod K)$th element of the K-dimensional key is used to construct the sub-trees at level $i+1$. For example, a 4D-Tree uses 4-dimensional vector $\langle a_0, a_1, a_2, a_3 \rangle$ as the key and a_0 is used to construct the sub-trees at level 1, 5, 9, $\ldots$, while a_1 is used to construct the sub-trees at level 2, 6, 10, etc. Figure 2.7 shows an example of a 3D-Tree.

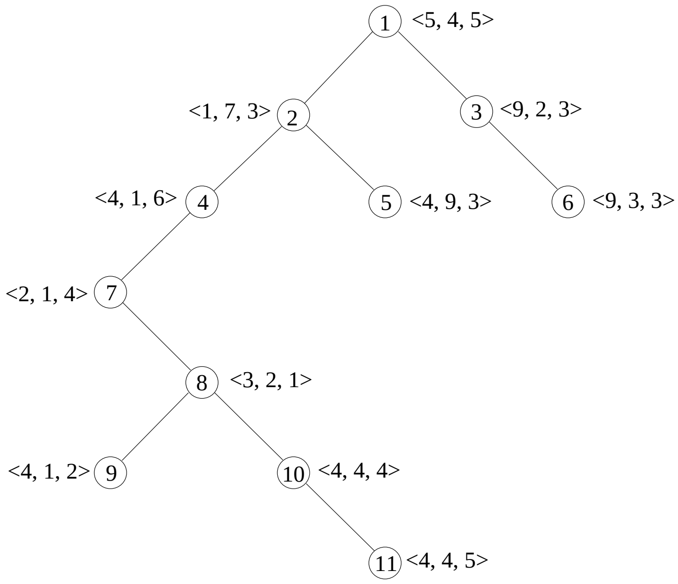


Figure 2.7: An example of 3D-Tree

The node number indicate the sequence where the nodes are inserted into the 3D-Tree and $\langle \cdot \rangle$ denotes the key at each node.

The normalization of NCS approach is as follows. Assume that a n-sided polygon is represented by an ordered list of n 2D points. A pair of points in the original list of coordinates is chosen to form a basis vector. A new coordinate system (the normalized one) is defined using the basis vector as a unit vector along the x-axis and each point in the original list is transformed to the new system. This process produces a list of normalized coordinates. Since the first two points of these normalized coordinates must be $(0, 0)$ and $(1, 0)$, they can be removed from the list. Thus, a n-sided polygon is represented by an ordered list of normalized coordinates with size of $n-2$. Figure 2.8 show the normalization process. The Euclidean distance of the normalized coordinate lists of two given polygons is used as the similarity measure between the two polygons. Two polygons are said to be matched if the Euclidean distance of their normalized coordinate lists is less than or equal to a predefined tolerance.

To maintain the index tree in practical size, only part of a normalized coordinate list, instead of the whole list, is used to construct the index tree. In [28, 29], experiments

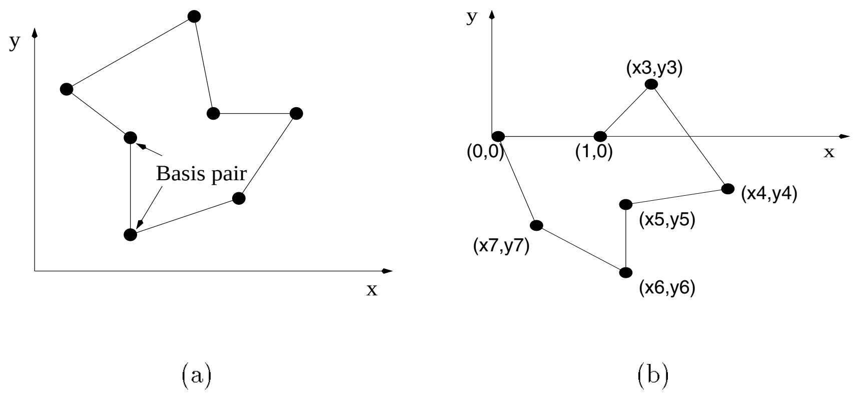


Figure 2.8: Normalizing coordinate list
(a) original coordinate list. (b) normalized coordinate list.

have been carried out using 4 points, 6 points, and 8 points. For a shape matching query, a number (the same as the number of points used to construct the index tree) of consecutive points in the normalized coordinate list of the target polygon are selected and used to traverse the index tree. All entries on the index tree matched with the query data points are selected and the polygons they are representing are regarded as hypothesis. The Euclidean distance testing is performed on these polygons and the target polygon to verify the hypothesis.

Characteristics

This approach is translation, scale and orientation invariant since the normalization is built into the approach itself. The storage complexity of this approach is $O(n)$ where n is the number of vertices of the polygons. The computational complexity for both the preprocessing and matching is also $O(n)$. Since this approach use the Euclidean Distance to measure the similarity between two polygons, it requires the two polygons to have the same number of vertices and a one-to-one vertex correspondence between the two polygons.

Discussion

Since only a portion of the normalized coordinate list is used, matches may be missed in the matching process. Consider a system which uses 4 points to organize the index tree. For a target polygon with n sides, where $n > 4$, 4 consecutive points are selected as the query data points to traverse the index tree. However, there may be no match found if the same 4 consecutive points are not selected for index construction when this polygon is inserted to the system. Hence, no hypothesis will be generated and the matching query fails. To solve this problem, [28, 29] propose to use more than one entries to represent a polygon when constructing the index tree. Consider the 4 points example again. For a n-gon where $n > 4$, there are n possible 4 consecutive point sequences since the normalized coordinate list is ordered. Therefore, n entries is inserted into the index tree when such a polygon is added to the system. Another problem of this approach is that it cannot handle similar queries efficiently. The reason is the same as that of the rectangular cover approach. And this problem can only be handle either by performing a linear search on the model polygons or an iterative method with tolerance relaxation as stated in Section 2.3.

2.6 The Hausdorff Distance Method

Idea

The Hausdorff Distance is a measurement of the distance between two point sets. It has been used for polygon matching and polygon similarity measuring tasks [20, 2]. Hausdorff Distance is defined as follows.

Definition 4 Given two finite point sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$, the Hausdorff Distance is

$$H(A, B) = \max(h(A, B), h(B, A))$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

and $\|\cdot\|$ is some underlying norm on the points of A and B.

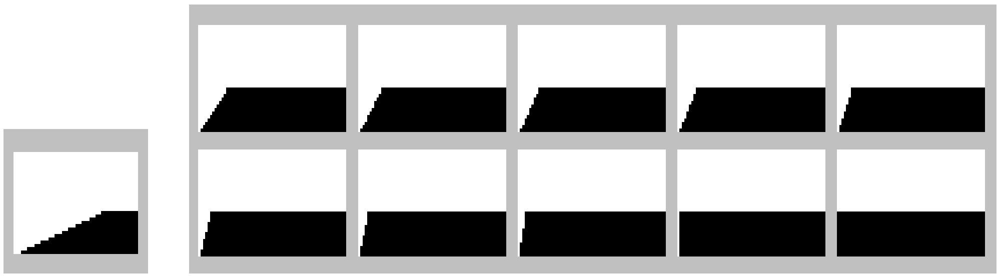
The mechanism of the Hausdorff Distance method is as follows. Each point in set A is paired up with the nearest point in set B. The distance of each pair of points is computed and the maximum of these distance is used to describe the similarity from set A to set B. The same procedure is applied from set B to set A. The larger one among the two values (set A to set B, and set B to set A) is used as the Hausdorff Distance between the two sets. In words, the Hausdorff Distance tells that for any point in a set, there is at least a point in another set that is within certain distance from it.

Characteristics

The Hausdorff Distance is translation, scale and orientation sensitive, polygons have to be normalized before they can be compared. The storage requirement and computational requirement are $O(n)$ and $O(n^2)$ respectively. This method is not indexable.

Discussion

The pairing found by the Hausdorff Distance method is not an one-to-one mapping. It means that some points in one set may be assigned to the same point in another set while some points in another set may not be assigned to any points at all. In such situations, the similarity measures between different polygons become the same. For example, the Hausdorff Distances between the polygon shown in Figure 2.9(a) and polygons shown in Figure 2.9(b) are all the same. This characteristic is undesirable when the Hausdorff Distance is used for polygon matching and it is the reason why this method produces polygon similarity measurements disagreed with human perception occasionally, as our experiments show (Chapter 7).



(a) (b)

Figure 2.9: Problem of Hausdorff Distance method

2.7 The PCA Approach

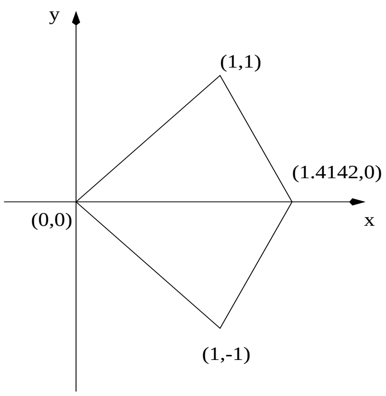
Idea

In [44], the authors proposed to perform fast pattern classification of attributed graphs using a PCA approach. One of the applications suggested in the paper is similar shape retrieval of polygonal shapes.

A polygon $P = \{v_1, v_2, \dots, v_n\}$ is represented by a attributed graph formed by the angle of vertex v_i to v_j in radian, i.e., $e_{i,j} = \arctan(v_i, v_j)$, $\forall v_i, v_j, i \neq j$, and $ei, j = \|v_i\|$. Figure 2.10(a) shows a polygon and the matrix of its attributed graph. After the attribute matrix is computed, the n eigenvalues of this matrix is computed and sorted in descending order. The similarity between two polygons is measured by the Euclidean Distance of the two sorted sets of eigenvalues of the polygons.

Characteristics

This approach is sensitive to translation, scale, and orientation. Thus, normalization is required before polygons are handled by this method. The storage requirement is $O(n)$. The computational complexity depends on the algorithm of computing the eigenvalues of the attribute matrix. In our implementation, we use the Jacobi method and the routine



0	$\frac{\pi}{4}$	0	$-\frac{\pi}{4}$
$-\frac{3\pi}{4}$	$\sqrt{2}$	$-\frac{\pi}{2}$	$-\frac{3\pi}{8}$
π	$\frac{\pi}{2}$	$\sqrt{2}$	$-\frac{5\pi}{8}$
$\frac{3\pi}{4}$	$\frac{5\pi}{8}$	$\frac{3\pi}{8}$	$\sqrt{2}$

(a) (b)

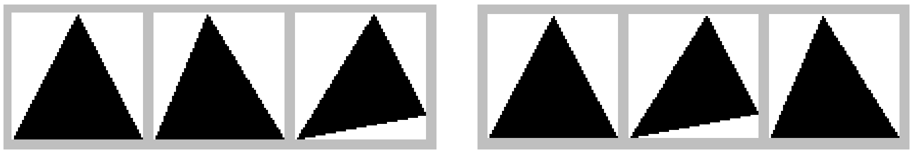
Figure 2.10: Attributed graph representation of a polygon

in [36] which runs in the time $O(n^3)$. This method does not require the knowledge of the correspondence between polygon vertices and the sorted eigenvalues can easily be indexed.

Discussion

The quality of the visual ranking produced by this approach is hard to analysis since it measure shape similarity using values produced by matrix decomposition. From empirical experiments, we found that this method occasionally produces undesirable visual ranking of shapes. For example, Figure 2.11(a) shows 3 polygons and Figure 2.11(b) shows the visual ranking produced by the PCA method when the first polygon in Figure 2.11(a) is used as the target polygon. The polygons in Figure 2.11(b) are placed from left to right in descending order of their similarity to the target polygon.

This approach can be made invariant to translation, scale, and orientation if we construct the attribute graph matrix with other attributes than edge directions. It is the $e_{i,j}, i \neq j$, entries that make the approach sensitive to orientation, and the $e_{i,i}$ entries that make the approach sensitive to translation and scale. Alternatively, we can have $e_{i,i} = 1$ and let $e_{i,j} = \|v_i - v_j\| / \max(e_{i,j}), i \neq j$, $\| \cdot \|$ denotes the Euclidean Distance. This will



(a) (b)

Figure 2.11: Undesirable visual ranking produced by PCA approach

makes the attribute graph matrix invariant to translation, scale, and orientation. However, quality of visual ranking produced the by this alternative version of PCA may be different from the original one as different shape attributes are used.

Table 2.1: Summary of the characteristics of different methods

	Freeman Chain Code	**Moment**	**Rectan-gular Cover**	**Potential-based**	**NCS**	**Hausdorff Distance**	**PCA**
Translation invariant	Yes	No	Yes	Yes	Yes	No	No‡
Scale invariant	No	No	Yes	Yes	Yes	No	No‡
Orientation invariant	No	No	No	Yes	Yes	No	No‡
Storage requirement	$O(k)$	$O(1)$	$O(l)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Computat-ional complexity	$O(k)$	$O(WH)$	$O(l)$	N/A†	$O(n)$	$O(n^2)$	$O(n^3)$
Require vertex correspond-ence	No	No	No	No	Yes	No	No
Indexable	Yes	Yes	Yes	No	Yes	No	Yes

l is the number of rectangles a rectangle cover.

k is the number of boundary pixels of a digitized shape.

n is the number of vertices of polygons.

W is the width of a frame buffer.

H is the height of a frame buffer.

† Complexity analysis not included in original paper.

‡ Yes, if distance between polygon vertices are used instead of angle.

Chapter 3

Binary Shape Descriptor

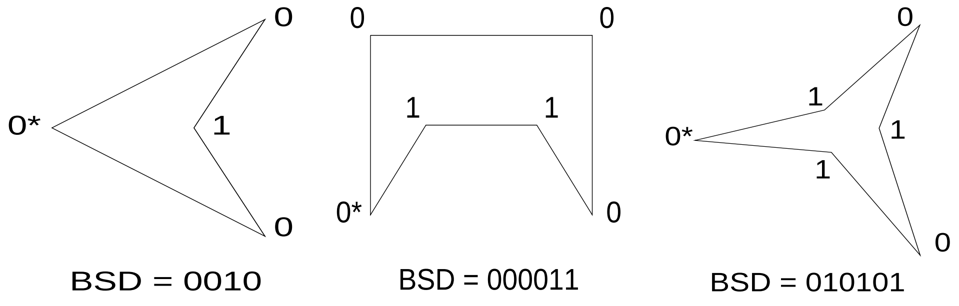
The BSD technique plays an important role in our two-stage framework. It serves as a polygon classification tool in our framework and is used to filter out potential dissimilar model shapes from the database and thus reduces the number of comparison needed for processing a shape query.

3.1 Basic idea

BSD is a binary string recording the convexities and concavities of the vertices of a polygon. Let '0' denotes a convex vertex (interior angle less than π) and '1' denotes a concave vertex (interior angle larger than π).

Definition 5 A Binary String Descriptor (BSD) is a string $a_1a_2 \ldots a_n$, where $a_i \in \{0, 1\}$ and n is the number of vertices of the polygon the descriptor.

Figure 3.1 shows several polygons and their corresponding BSDs. BSD is scale and orientation invariant since the measurement of convexity and concavity of a vertex is independent of these properties. However, the specific instance of the BSD of a polygon depends on the selection of the *anchor vertex* (the vertex of the polygon at which we start recording the BSD).



(a) (b) (c)

Figure 3.1: Polygons and their BSDs

The anchor vertices are marked with *. BSDs are recorded clockwise.

3.2 Standardized Binary String Descriptor

A polygon can be represented by more than one BSD depending on the sequence of vertices being recorded. For example, a polygon represented by BSD "0010" can also be represented by "0100", "1000" or "0001", depending on the anchor vertex (Figure 3.2). The idea of standardized BSD is introduced in [6] in order to obtain a unique BSD for a given polygon.

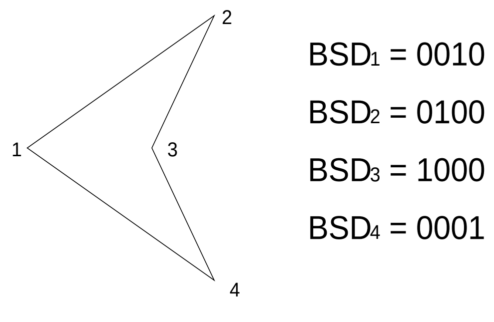


Figure 3.2: Different BSDs for the same polygon

$BSD_{1...4}$ are the BSDs for the same polygon which are recorded with different anchor vertex $1 \ldots 4$.

Definition 6 Given a BSD $B = \{0,1\}^n$, a rotated BSD B_i, for $1 \le i \le n$, is another BSD generated by rotating the bits of B such that the ith Most Significant Bit (MSB) of B becomes the MSB of B_i.

Definition 7 Let $M(B_i)$ denotes the magnitude of B_i when it is treated as a binary integer. The Standardized Binary String Descriptor (SBSD) of B is B_j such that

$$M(B_j) = \min_i M(B_i), 1 \le i \le n$$

For example, given a BSD B="0010", B_1, B_2, B_3 and B_4 are "0010", "0100", "1000" and "0001" respectively. Since B_4 is the one with the smallest magnitude, SBSD of $B = B_4$ = "0001".

SBSD inherits the scale and orientation invariant properties from BSD and it is independent of the selection of anchor vertex.

3.3 Number of equivalent classes for n-gons

SBSD function is a many-to-one mapping, i.e. more than one polygon may have the same SBSD. For example, the SBSD of the polygons in Figure 3.3 are all "00001". Polygons having the same SBSD are said to be in the same equivalent class. For polygons with n sides, there are 2^n possible BSDs. However, some of them are invalid. For example, '00111' is invalid since a simple polygon should have at least three convex vertices, thus all valid BSD should have at least three '0's. Some BSDs are the same after standardization, for example, '00011', '01100' and '10001'. For n-gons, the number of equivalent classes (E) is given in [6] as

$$E = \frac{1}{n} \sum_{m \in D_n} m X_n(m) - (\lfloor \frac{n}{2} \rfloor + 2)$$

where D_n is the set of divisors of n,

$$X_n(m) = 2^{\frac{n}{m}} - (X_n(m_1) + \cdots + X_n(m_k))$$

and $m_1, \cdots, m_k$ are the multiples of m belonging to $D_n \backslash \{m\}$.

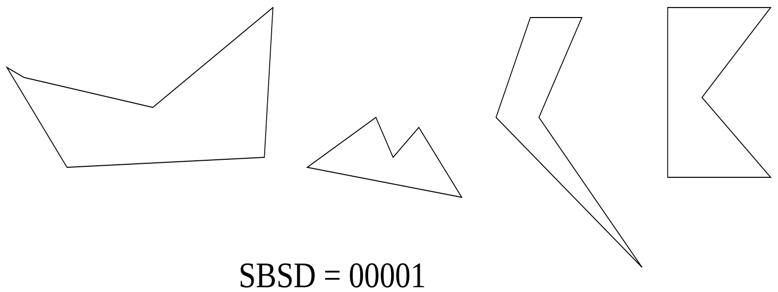


Figure 3.3: Polygon in an equivalent class

Table 3.1 shows the number of equivalent classes for polygons with sides from 3 to 16, where n is the number of polygon vertices, E is the number of equivalent classes, and $\overline{E} = \frac{2^n}{E}$ shows the average number of BSDs in an equivalent class. From the table, we find that the number of equivalent classes is small when n is small. This is an undesirable characteristic since it means a lot of polygons will be group into the same equivalent class when n is small, and thus reduce the pruning effect of the two-stage framework. A possible extension to tackle this problem is discussed in Chapter 8. We also observe that $\overline{E}$ increases when n increases. It shows that the average number of BSDs in an equivalent class increases with n.

Table 3.1: N-gons and number of their equivalent classes

$\boldsymbol{n}$	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$\boldsymbol{2^n}$	8	16	32	64	128	256	512
$\boldsymbol{E}$	1	2	4	9	15	30	54
$\boldsymbol{\overline{E}}$	8	8	8	7.11	8.53	8.53	9.48
$\boldsymbol{n}$	**10**	**11**	**12**	**13**	**14**	**15**	**16**
$\boldsymbol{2^n}$	1024	2048	4096	8192	16384	32786	65536
$\boldsymbol{E}$	101	181	343	624	1173	2183	4106
$\boldsymbol{\overline{E}}$	10.14	11.31	11.94	13.13	13.97	15.01	15.96

Chapter 4

The Two-Stage Framework

We propose a two-stage framework for polygon retrieval which incorporates both qualitative and quantitative measures of polygons in the first and the second stage respectively. The main idea of the framework is to partition model polygons into groups according to their SBSDs. Instead of comparing the target polygon with every model polygons, we only compare it with the model polygons having SBSDs within a certain Hamming Distance tolerance from the SBSD of the target polygon. The selected model polygons are then passed to the second stage of the framework for quantitative similarity measure.

Given two binary strings $A = a_1, a_2, \ldots, a_n$ and $B = b_1, b_2, \ldots, b_n$, the Hamming Distance between A and B is defined as

$$HAMDIST(A, B) = \sum_{i=1}^{n} c_i$$

where

$$c_i = \begin{cases} 0 & \text{if } a_i = b_i \\ 1 & \text{if } a_i \neq b_i \end{cases}$$

By using the Hamming Distance tolerance, the two-stage framework provides a systematic way for controlling the degree of search space pruning by allowing user to make tradeoff between quality of query result and speed of query processing. A small tolerance produces larger pruning effect by selecting fewer model polygons for comparison but with higher risk of excluding potential candidates and thus resulted in worse query results.

On the other hand, large tolerance may produce better query results but has a smaller pruning effect.

Figure 4.1 shows three polygons with their SBSDs. If we use polygon P as the target polygon and specify a Hamming Distance tolerance of 0, then only polygon P and polygon R will be selected at the first stage of the framework. However, if we specify a Hamming Distance tolerance of 1, then all polygon P, Q, and R will be selected at the first stage of the framework.

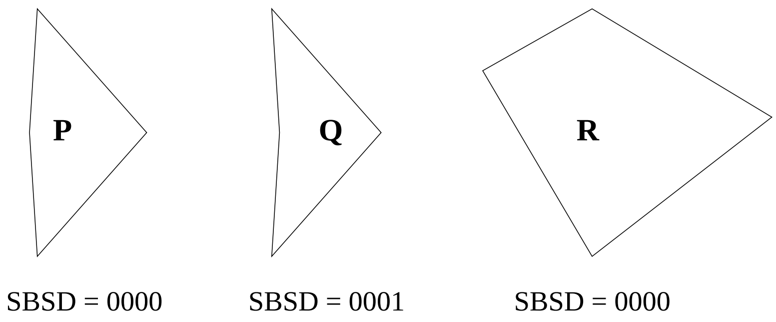


Figure 4.1: An example of polygon selection at the first stage of the two-stage framework

The second stage of the framework incorporates any available polygon matching and similarity measuring techniques. For handling matching queries, the technique incorporated only need to determine whether two given polygons match each other under user specified tolerance. However, in order to handle similarity queries, the technique incorporated must be able to produce a ranking on the set of model polygons according to the degree of similarity between these polygons and the target polygons. One possible way of accomplishing this task is to incorporate polygon similarity measuring techniques which produce a numerical value as the similarity measure between two polygons. The ranking can then be produced by sorting the model polygons based on their similarity measure to the target polygon.

Putting the two stages together, the framework should look like:

1. $\boldsymbol{Q} = \{P_i \mid P_i \in \boldsymbol{P} \ \wedge \ HAMDIST(SBSD(P_i), SBSD(T)) \leq \delta\}$ where $\boldsymbol{Q}$ is the set of polygons selected for second stage matching, $\boldsymbol{P}$ is the set of model polygons, T is the target polygon, $SBSD(\cdot)$ denoted the SBSD function, $HAMDIST(\cdot)$ denotes

the function that computes the Hamming Distance between two binary strings, and δ is the user specified Hamming Distance tolerance.

2. If the query is an matching query, execute (a). If it is a similarity query, execute (b).

   (a) $\boldsymbol{R} = \{Q_i \mid Q_i \in \boldsymbol{Q} \ \wedge \ MATCH(Q_i, T)\}$ where $MATCH(\cdot)$ denotes the polygon matching method selected.

   (b) $\boldsymbol{R} = \{Q_i \mid Q_i \in \boldsymbol{Q} \ \wedge \ 1 \leq i \leq n\}$ where n is the number of model polygons to be included in the answer to a query, m is the number of model polygons in a database, $n \leq m$, $\boldsymbol{Q} = \{Q_1, Q_2, \ldots, Q_m\}$, and $Q_1 \leq Q_2 \leq \cdots \leq Q_m$, which is the ranking produced by the polygon similarity measuring technique selected.

3. $\boldsymbol{R}$ is the set of model polygons which is the answer to the query.

This two-stage framework has three main advantages comparing to model-driven approach and data-driven approach:

1. The two-stage framework is more efficient than traditional model-driven approach since it reduces the number of comparison by filtering out model shapes that are potentially dissimilar to the target shape.

2. The two-stage framework avoids the difficulty and inefficiency of maintaining complex multi-dimensional index structures in data-driven approach. It also avoids the extra index maintenance work during database modification and index reorganization.

3. Using the Hamming Distance tolerance, the two-stage framework provides user with systematic way for controlling the degree of search space pruning and quality of query result.

Chapter 5

Multi-Resolution Area Matching

In this section, we propose a multi-resolution area-based polygon similarity measuring technique. We describe how the Multi-Resolution Area Information (MRAI) is computed and how similarity between polygons is measured using MRAI. We also describe how to take advantage of the multi-resolution characteristic of this technique to further speed up shape query processing.

5.1 The idea

The main idea of the MRAM technique is to measure the area difference between polygons in a multi-resolution manner. In this method, the area information of a polygon at several predefined resolutions are computed and stored using the Quadtree [37] approach. We then use the stored area information of two polygons to compute the area difference between the polygons from coarse resolution to fine resolution.

The advantage of using a multi-resolution approach is that computation can be saved by introducing the concept of early rejection. As we described in Section 5.2, the size of the area information at a coarse resolution is smaller than the size of the area information at a fine resolution. Thus, the computational complexity for computing the area difference between two polygons at a coarse resolution is less than the one at a fine resolution. As the two polygons are compared from coarse resolution to fine resolution, the comparison can be terminated as soon as the area difference between the two polygons are large enough to conclude that they are dissimilar. If dissimilarity are found at the coarse resolution, we can save a lot of computation by skipping unnecessary comparison at finer resolutions.

5.2 Computing MRAI

A polygon is first rotated such that the longest edge is aligned with y-axis and is normalized to have a unit bounding box. After the normalization, it is scan-converted onto a frame buffer with $W \times W$ pixels. MRAI is then computed using a Quadtree like approach:

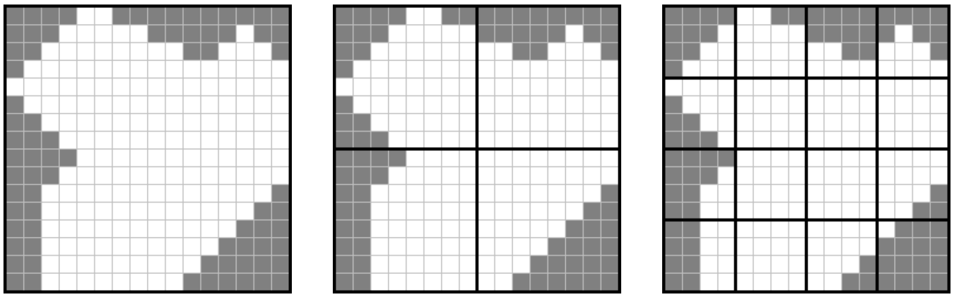
1. MRAI is recorded starting at level 0.
2. At level 0, the whole frame buffer is regarded as a cell. The portion of area covered by the polygon is recorded.
3. At level k, cells are obtained by quartering every cell of level $k-1$. The portion of area covered by the polygon in each level-k cell is recorded. There are 4^k cells at level k.

The MRAI at each level is concatenated to form a complete MRAI vector. The size of this vector, L, is

$$L = \sum_{i=0}^{K} 4^i = \frac{4^{K+1}-1}{3}$$

where K is the maximum resolution level of MRAI to be recorded.

Figure 5.1 shows an example of computation of MRAI. In this example, the size of the frame buffer is 16 pixels by 16 pixels large and MRAI is recorded up to resolution level 2. Figure 5.1(a) shows the cell partitioning of resolution level 0. The area information recorded at level 0 is $\langle \frac{180}{256} \rangle$. Likewise, Figure 5.1(b) shows the cell partitioning of resolution level 1 and the area information recorded at level 1 is $\langle \frac{46}{64}, \frac{46}{64}, \frac{45}{64}, \frac{43}{64} \rangle$. The same operation is applied at resolution level 2. By concatenating the area information of the three levels, we obtain the complete area information as: $\langle \frac{180}{256}, \frac{46}{64}, \frac{46}{64}, \frac{45}{64}, \frac{43}{64}, \frac{6}{16}, \frac{14}{16}, \frac{6}{16}, \frac{8}{16}, \frac{10}{16}, \frac{16}{16}, \frac{16}{16}, \frac{16}{16}, \frac{5}{16}, \frac{16}{16}, \frac{16}{16}, \frac{13}{16}, \frac{8}{16}, \frac{16}{16}, \frac{13}{16}, \frac{1}{16} \rangle$.



(a) (b) (c)

Figure 5.1: Computing Multi-Resolution Area Information (MRAI)
(a) Resolution level 0 with 1 cell. (b) Resolution level 1 with 4 cells. (c) Resolution level 2 with 16 cells.

In our implementation, the frame buffer is 64 pixels by 64 pixels large and we record area information up to resolution level 3 so the size of a complete MRAI vector is 85. Algorithm 5.1 shows the computation of MRAI in our implementation.

Algorithm 5.1 Computing multi-resolution area information

/* *Inputs:* $A[0..W-1][0..W-1]$ *is a 2D bit array containing the scan-converted polygon which a '1' indicates a pixel covered by the polygon;* W *is the width of the frame buffer;* K *is the maximum resolution level at which area information is to be recorded.* */

/* *Output:* $H[0..L-1]$ *is an array containing the multi-resolution area information where* $L = \frac{4^{K+1}-1}{3}$ */

$s \leftarrow \frac{W}{2^K}$
for $y = 0$ **to** $2^K - 1$ **do**
  for $x = 0$ **to** $2^K - 1$ **do**
    $idx \leftarrow \frac{4^K-1}{3} + y \cdot 2^K + x$
    $H[idx] \leftarrow \sum_{p=0}^{s-1} \sum_{q=0}^{s-1} A[y \cdot s + p][x \cdot s + q]$

$H[idx] \leftarrow \frac{H[idx]}{s^2}$

end for

end for

for $l = K$ **downto** 1 **do**

for $y = 0$ **to** $2^{l-1} - 1$ **do**

for $x = 0$ **to** $2^{l-1} - 1$ **do**

$idx \leftarrow \frac{4^{l-1}-1}{3} + y \cdot 2^{l-1} + x$

$idx_1 \leftarrow \frac{4^l-1}{3} + 2 \cdot y \cdot 2^l + 2 \cdot x$

$idx_2 \leftarrow \frac{4^l-1}{3} + 2 \cdot y \cdot 2^l + (2 \cdot x + 1)$

$idx_3 \leftarrow \frac{4^l-1}{3} + (2 \cdot y + 1) \cdot 2^l + 2 \cdot x$

$idx_4 \leftarrow \frac{4^l-1}{3} + (2 \cdot y + 1) \cdot 2^l + (2 \cdot x + 1)$

$H[idx] \leftarrow \frac{1}{4} \cdot \sum_{i=1}^{4} idx_i$

end for

end for

end for

return H

5.3 Measuring similarity using MRAI

We use the L_p distance to measure the similarity of two polygons at a specific level of resolution. Given polygon A and B, with their MRAI, the similarity of these two polygons at resolution level k is:

$$S_k(A, B) = (\sum_{i=1}^{4^k} |A_{ki} - B_{ki}|^p)^{\frac{1}{p}}$$

where $S_k(A, B)$ is the similarity measure of A and B at resolution level k, A_{ki} and B_{ki} are the portion of covered area in level k cells of polygon A and B respectively. In our implementation, we use $p = 2$, which is the Euclidean Distance.

Definition 8 Two polygons A and B are said to be matched at level k if $S_k(A, B) \leq \delta_k$ where δ_k is a predefined threshold value for level k similarity measure.

Definition 9 Two polygons are said to be matched if they are matched at all levels, i.e. the two polygons are similar at level $0, \cdots, K$.

Note that the choice of parameter p can be made based on user's tolerance on local dismatches between shapes. Large p values tend to emphasize local dismatches between shapes during the matching process. Table 5.1 shows 3 MRAIs at resolution level 1. (b) is differ from (a) in a uniform manner in terms of local dismatches of cells. (c) matches (a) well but with a relatively large dismatch in one of its cells.

Table 5.1: 3 MRAIs at resolution level 1

0.5	0.5
0.5	0.5

(a)

0.6	0.6
0.6	0.6

(b)

0.5	0.7
0.5	0.5

(c)

Table 5.2 shows the similarity measure $\delta_1(\cdot)$ when different values of p are used. When $p = 1$, (c) is considered to be more similar to (a) than (b) does. When $p = 2$, (c) and (b) are considered equally similar to (a). When $p \geq 3$, (c) is considered less similar to (a) than (b) does. This example shows that local dismatches between shapes are emphasized when the value of p increases. Thus, small p values have larger tolerance of local dismatches and tend to evaluate the similarity between shapes in a relatively global view. Large p values are relatively sensitive to local dismatches so they should not be used if the shapes to be processed are obtained from noisy channels.

Matching of two polygons can be done in levels, that is, perform similarity measuring from coarse resolution (level 0) to fine resolution (the maximum resolution level K). Algorithm 5.2 shows our implementation of polygon matching using MRAI:

Algorithm 5.2 Polygon matching using MRAI

Table 5.2: MRAM similarity measures with different p values

$\boldsymbol{p}$	$\boldsymbol{\delta_1(a,b)}$	$\boldsymbol{\delta_1(a,c)}$
1	0.4	0.2
2	0.2	0.2
3	0.1587	0.2
4	0.1414	0.2

/* *Inputs: K is the maximum resolution level at which area information is recorded; $\delta_{0..K}$ are the predefined thresholds for similarity measure at resolution level 0 to K; p is the parameter of the L_p function to be used; $H_1[0..L-1]$ and $H_2[0..L-1]$ are the MRAI of the two polygons and $L = \frac{4^{K+1}-1}{3}$.* */
/* *Output: A boolean value (**true** or **false**) indicating whether the two given polygons are matched or not.* */

for $l = 0$ **to** K **do**
  $idx_1 \leftarrow \frac{4^l-1}{3}$
  $idx_2 \leftarrow idx_1 + 4^l - 1$
  $S \leftarrow (\sum_{i=idx_1}^{idx_2} |H_1[i] - H_2[i]|)^{\frac{1}{p}}$
  if $(S > \delta_l)$ **then**
    return false
  end if
end for
return true

5.4 Query processing using MRAM

Shape query processing can be further speeded up by taking advantage on MRAM's multi-resolution characteristic. The main idea is to locate those model polygons which are dissimilar to the target polygon in coarse resolution level. Since in MRAM, polygons are compared from coarse resolution level to fine resolution level and the computational cost for comparison in coarse resolution is less than that of fine resolution level, early rejection of dissimilar model polygons will reduce the computation needed. We describe the procedures for handling matching queries and similarity queries using MRAM. Note that the model polygons mentioned in the following two sessions are those selected by the first stage algorithm of the two-stage framework according to the target polygon.

Matching query

Since level 0 MRAI is just a numerical value, it can serve as a database index for the model polygons. Matching queries are processed as follows.

1. Only model polygons having level 0 MRAI in the range $[t_0 - \delta_0, t_0 + \delta_0]$ are fetched for further matching where t_0 is the level 0 MRAI of the target polygon and δ_0 is the predefined threshold for level 0 matching as stated in Section 5.3. Since model polygons are indexed on level 0 MRAI, this subset of polygons can be retrieved efficiently.

2. Model polygons selected in the previous step are then compared with the target polygon individually using the approach described in Section 5.3.

Similarity query

Similarity queries are processed using following algorithm

$\boldsymbol{Q} \leftarrow$ model polygons
for $i = 0$ **to** K **do**
  sort $\boldsymbol{Q}$ in descending order of $S_i(Q_j, T)$ where $Q_j \in \boldsymbol{Q}$
  $\boldsymbol{Q} \leftarrow \{Q_j \mid Q_j \in \boldsymbol{Q} \wedge 1 \leq j \leq n_i\}$
end for

where T is the target polygon and $S_i(\cdot)$ is the level i similarity measure of two polygons as stated in Section 5.3. In our implementation, n_0, n_1, n_2 and n_3 are 100, 50, 25 and 10 respectively.

5.5 Characteristics and Discussion

The MRAM method is sensitive to translation, scale, and orientation so polygons have to be normalized (Section 7.1.2) before they are handled by MRAM method. The storage requirement is $O(L)$ where L is the size of the MRAI which depends on the maximum resolution level, K, used and $L = \sum_{i=0}^{K} 4^i = \frac{4^{K+1}-1}{3}$. The computational requirement is also $O(L)$.

The MRAM method is not fully indexable. One of the reasons is that the size of the MRAI extracted is rather large so that it is hard to be indexed efficiently. For example, the MRAI is a 85-dimensional vector if the maximum resolution level is 3. The fact that the matching and similarity measuring processes are iterative also makes it difficult to index the MRAI. Though the whole MRAI can not be practically indexed, we can index part of it, e.g. only index the first few resolution levels of MRAI as stated in Section 5.4. This will definitely improves the performance of the MRAM method.

Chapter 6

Circular Error Bound and Minimum Circular Error Bound

In this session, we introduce a polygon matching technique using Circular Error Bound (CEB) which can be used to determine whether two given polygon resemble each other within some predefined tolerance. This technique can be used to handle matching queries. We extend the idea of CEB and introduce a polygon similarity measuring technique named Minimum Circular Error Bound (MCEB). This method gives a translation invariant measurement on the degree of resemblance between two given polygons. It can be used to handle similar queries.

6.1 Polygon Matching using Circular Error Bound

The polygon matching technique using CEB is based on an intuitive human perception of polygon resemblance. The intuitive perception of similar polygons is as follows. If two polygons are matched, then each vertex of one polygon should be close to its corresponding vertex of another polygon when the two polygons are overlapped. The correspondence between vertices is an one-to-one mapping. Therefore, the definition and the technique we proposed only work on polygons that have the same number of vertices. Before the two polygons are overlapped, translation, scaling and rotation are allowed to be performed on the polygons.

Definition 10 A transformation T is a vector, i.e. $T = \langle t_x, t_y, s_x, s_y, \theta \rangle$ where t_x is translation in x-axis direction, t_y is translation in y-axis direction, s_x is the scaling in x-axis direction, s_y is the scaling in y-axis direction and θ is the rotation about the origin. $T(Q)$ denotes the object obtained by applying T to Q where Q may be a polygon or a vertex.

Definition 11 Given a tolerance vector $E = \langle \epsilon_1, \epsilon_2, \ldots, \epsilon_n \rangle$, $Q = \{U_1, U_2, \ldots, U_n\}$ is said to be matched with $P = \{V_1, V_2, \ldots, V_n\}$ if there exists a transformation T such that $Q' = T(Q) = \{U'_1, U'_2, \ldots, U'_n\}$ and $\forall_{1 \leq i \leq n} \|V_i - U'_i\| \leq \epsilon_i$, where $\|\cdot\|$ denotes the Euclidean norm *(Figure 6.1)*.

Definition 12 Given V_i, ϵ_i and U_i, the ith *Circular Error Bound (CEB)* , C_i, is a circle with ϵ_i as its radius and $(V_i - U_i)$ as its center.

Note that Definition 11 assumes we already know the pairing of vertices between the two polygons, i.e. V_i should match U_i.

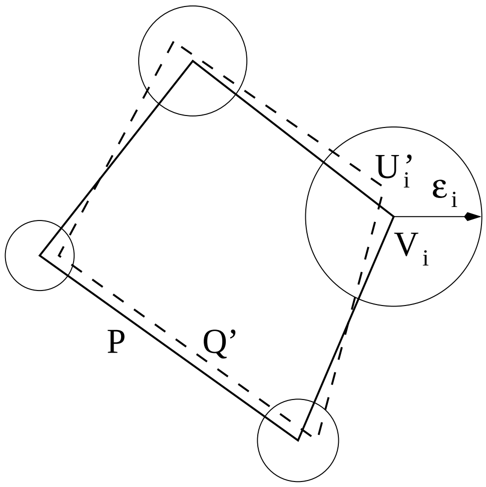


Figure 6.1: An intuitive definition of similar polygons

The polygon matching task using CEB is formulated as follows:

"Given two polygons P and Q with a tolerance vector E, the task is to determine whether a transformation T exists such that Q is said to be matched with P under Definition 11."

By Definition 11, the transformation T is an arbitrary vector $\langle t_x, t_y, s_x, s_y, \theta\rangle$. However, in nowadays applications, the transformations in polygon matching task are often restricted to some special cases, for example, translation and (or) scaling only. With restricted transformations, we have efficient solutions for the polygon matching task. In the following sections, we will present the solution for the polygon matching task when

1. only translations are allowed
2. only translations and uniform scaling in x-axis and y-axis directions are allowed
3. only translations and independent scaling in x-axis and y-axis directions are allowed.

6.1.1 Translation

Assume that the transformation T in Definition 11 is restricted to $T = \langle t_x, t_y, 1, 1, 0\rangle$ only.

Proposition 1 Given $P = \{V_1, V_2, \dots, V_n\}$, $Q = \{U_1, U_2, \dots, U_n\}$, $E = \langle \epsilon_1, \epsilon_2, \dots, \epsilon_n\rangle$, Q is matched with P if and only if the n Circular Error Bounds $C_1, C_2, \dots, C_n$ of P and Q have common intersection (Figure 6.2).

Proof 1

If : Assuming $V_i = (a_i, b_i)$ and $U_i = (c_i, d_i)$, by Definition 12, Circular Error Bound C_i is a circle with ϵ_i as its radius and $(a_i - c_i, b_i - d_i)$ as its center. If $C_1, C_2, \dots, C_n$ have common intersection, then for any point (t_x, t_y) in the common intersection, the distance between this point and the center of any C_i is less than or equal to the radius of C_i. Figure 6.2 illustrates this idea when both P and Q are triangles. Thus, $\forall i, 1 \le i \le n$,

$$\sqrt{[(a_i - c_i) - t_x]^2 + [(b_i - d_i) - t_y]^2} \le \epsilon_i \tag{6.1}$$

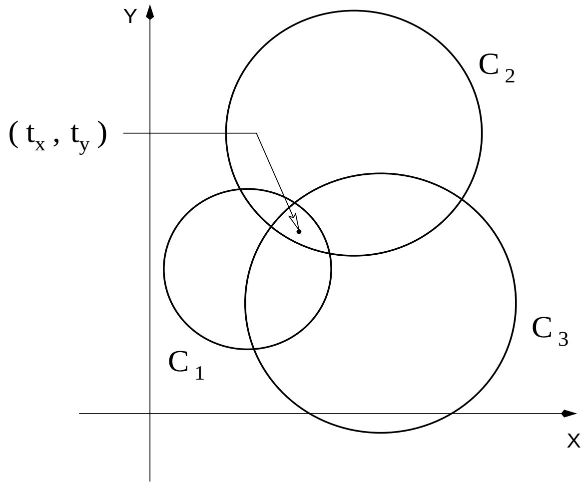


Figure 6.2: Intersection of Circular Error Bounds

Re-arranging Equation 6.1, we have

$$\sqrt{[a_i - (c_i + t_x)]^2 + [b_i - (d_i + t_y)]^2} \leq \epsilon_i \tag{6.2}$$

which is equivalent to $\|V_i - U_i'\| \leq \epsilon_i$ where $U_i' = T(U_i)$ and $T = \langle t_x, t_y, 1, 1, 0\rangle$. By Definition 11, Q is matched with P.

Only if : Assume that $V_i = (a_i, b_i)$ and $U_i = (c_i, d_i)$. By Definition 11, if Q is matched with P then $\exists T = (t_x, t_y, 1, 1, 0)$ such that $\forall_{1\leq i\leq n}, [a_i - (c_i + t_x)]^2 + [b_i - (d_i + t_y)]^2 \leq \epsilon_i^2$. Let $\alpha = a_i - (c_i + t_x)$, $\beta = b_i - (d_i + t_y)$, $\gamma = a_j - (c_j + t_x)$, and $\delta = b_j - (d_j + t_y)$, we have $\alpha^2 + \beta^2 \leq \epsilon_i^2$, $\gamma^2 + \delta^2 \leq \epsilon_j^2$, and

$$
\begin{aligned}
0 &\leq (\alpha\delta - \beta\gamma)^2 \\
2\alpha\beta\gamma\delta &\leq \alpha^2\delta^2 + \beta^2\gamma^2 \\
\alpha^2\gamma^2 + \beta^2\delta^2 + 2\alpha\beta\gamma\delta &\leq \alpha^2\gamma^2 + \alpha^2\delta^2 + \beta^2\gamma^2 + \beta^2\delta^2 \\
(\alpha\gamma + \beta\delta)^2 &\leq (\alpha^2 + \beta^2)(\gamma^2 + \delta^2) \\
(\alpha\gamma + \beta\delta)^2 &\leq \epsilon_i^2\epsilon_j^2 \\
-\alpha\gamma - \beta\delta &\leq \epsilon_i\epsilon_j \\
-2\alpha\gamma - 2\beta\delta &\leq 2\epsilon_i\epsilon_j \\
\alpha^2 + \beta^2 + \gamma^2 + \delta^2 - 2\alpha\gamma - 2\beta\delta &\leq \epsilon_i^2 + \epsilon_j^2 + 2\epsilon_i\epsilon_j \\
(\alpha - \gamma)^2 + (\beta - \delta)^2 &\leq (\epsilon_i + \epsilon_j)^2 \\
\{[a_i - (c_i + t_x)] - [a_j - (c_j + t_x)]\}^2 + & \\
\{[b_i - (d_i + t_y)] - [b_j - (d_j + t_y)]\}^2 &\leq (\epsilon_i + \epsilon_j)^2 \\
[(a_i - c_i) - (a_j - c_j)]^2 + [(b_i - d_i) - (b_j - d_j)]^2 &\leq (\epsilon_i + \epsilon_j)^2 \\
\sqrt{[(a_i - c_i) - (a_j - c_j)]^2 + [(b_i - d_i) - (b_j - d_j)]^2} &\leq (\epsilon_i + \epsilon_j)^2 \qquad (6.3)
\end{aligned}
$$

Because $(a_i - c_i, b_i - d_i)$, $(a_j - c_j, b_j - d_j)$ are the centers of Circular Error Bound C_i and C_j respectively, and ϵ_i, ϵ_j are their radius, C_i and C_j intersect each other. Therefore, $\forall_{1 \leq i,j \leq n}$, C_i and C_j intersect each other. Thus, $C_1, C_2, \ldots, C_n$ have common intersection.

6.1.2 Translation and uniform scaling in x-axis and y-axis directions

Assume that the transformation T in Definition 11 is restricted to $T = \langle t_x, t_y, s, s, 0\rangle$, i.e. only translation and uniform scaling in x-axis and y-axis directions are allowed.

Let $V_i = (a_i, b_i)$ and $U_i = (c_i, d_i)$. Consider the following equation

$$
\begin{aligned}
&[(c_i - c_j)^2 + (d_i - d_j)^2]s^2 - 2[(a_i - a_j)(c_i - c_j) + (b_i - b_j)(d_i - d_j)]s \\
&\quad + [(c_i - c_j)^2 + (d_i - d_j)^2 - (\epsilon_i + \epsilon_j)^2] \leq 0 \qquad (6.4)
\end{aligned}
$$

Solving Equation 6.4, we get a range, $\mathbb{S}_{ij}$, for s that the inequality holds.

Proposition 2 Q is matched with P if and only if $\bigcap_{1\leq i,j\leq n} \mathbb{S}_{ij} \neq \emptyset$ (Figure 6.3).

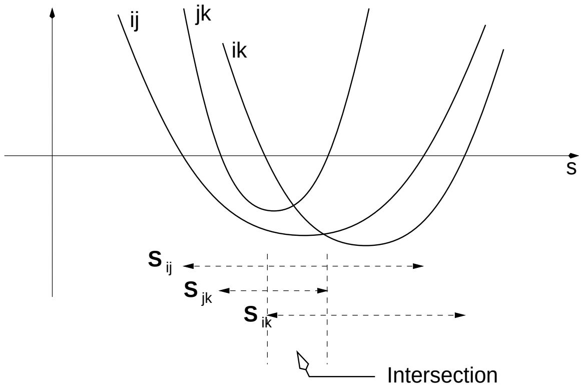


Figure 6.3: $\mathbb{S}_{ij}$ and its intersection

Proof 2

If : If $\bigcap_{1\leq i,j\leq n} \mathbb{S}_{ij} \neq \emptyset$, then $\exists S = \langle 0, 0, s, s, 0\rangle \in \bigcap_{1\leq i,j,\leq n} \mathbb{S}_{ij}$ such that Circular Error Bounds $C_1, C_2, \ldots, C_n$ of P and Q' have common intersection, where $Q' = S(Q)$. By Proposition 1, Q' is matched with P. Thus, $\exists T = \langle t_x, t_y, 1, 1, 0\rangle$ such that $\forall_{1\leq i\leq n} \|V_i - U''_i\| \leq \epsilon_i$ where $U''_i = T(U'_i)$. Therefore, $\exists T' = T\circ S = \langle t_x, t_y, s, s, 0\rangle$ such that $\forall_{1\leq i\leq n}\|V_i - U''_i\| \leq \epsilon_i$ where $U''_i = T'(U_i)$. By Definition 11, Q is matched with P.

Only if : By Definition 11, if Q is matched with P then $\exists T = (t_x, t_y, s, s, 0)$ such that $\forall_{1\leq i\leq n}, [a_i - (sc_i + t_x)]^2 + [b_i - (sd_i + t_y)]^2 \leq \epsilon_i^2$. As shown in Proof 1, the n Circular Error Bounds $C_1, C_2, \ldots, C_n$ have common intersection. Thus, we have $\forall_{1\leq i,j\leq n}$

$$\sqrt{[(a_i - sc_i) - (a_j - sc_j)]^2 + [(b_i - sd_i) - (b_j - sd_j)]^2} \leq \epsilon_i + \epsilon_j \qquad (6.5)$$

Re-arranging Equation 6.5, we have Equation 6.4. Thus, $\exists s$ such that $\forall_{1\leq i,j\leq n}$, Equation 6.4 holds. Therefore, if $\mathbb{S}_{ij}$ denotes the range of s where Equation 6.4 holds for a specific ij pair, $\bigcap_{1\leq i,j\leq n} \mathbb{S}_{ij} \neq \emptyset$.

6.1.3 Translation and independent scaling in x-axis and y-axis directions

Assume that the transformation T in Definition 11 is restricted to $T = \langle t_x, t_y, s_x, s_y, 0\rangle$, i.e. independent scaling in x-axis and y-axis directions as well as translation are allowed.

Let $V_i = (a_i, b_i)$ and $U_i = (c_i, d_i)$. Consider the following equation

$$[(a_i - a_j) - (c_i - c_j)s_x]^2 + [(b_i - b_j) - (d_i - d_j)s_y]^2 \leq (\epsilon_i + \epsilon_j)^2 \tag{6.6}$$

Equation 6.6 defines an ellipse, $\mathbb{E}_{ij}$, on the s_x-s_y plane.

Proposition 3 Q is matched with P if and only if $\forall_{1\leq i,j\leq n}\mathbb{E}_{ij}$ have common intersection (Figure 6.4).

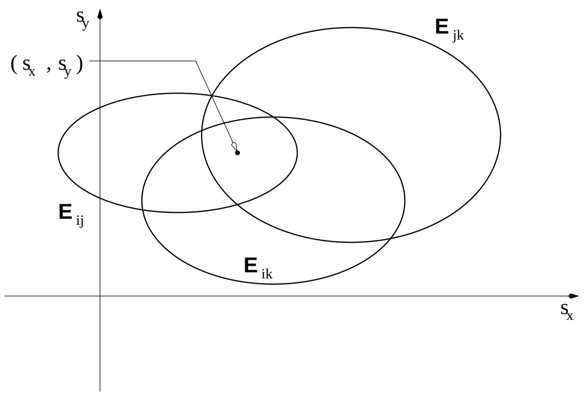


Figure 6.4: $\mathbb{E}_{ij}$ and its intersection

Proof 3

If : If $\forall_{1\leq i,j\leq n}\mathbb{E}_{ij}$ have common intersection, then for any point (s_x, s_y) in the common intersection, Circular Error Bounds $C_1, C_2, \ldots, C_n$ of of P and Q', intersect each other, where $Q' = S(Q)$ and $S = \langle 0, 0, s_x, s_y, 0\rangle$. By Proposition 1, Q' is matched with P. Thus, $\exists T = \langle t_x, t_y, 1, 1, 0\rangle$ such that $\forall_{1\leq i\leq n}, \|V_i - U_i''\| \leq \epsilon_i$ where $U_i'' = T(U_i')$. Therefore, $\exists T' = T \circ S = \langle t_x, t_y, s_x, s_y, 0\rangle$ such that $\forall_{1\leq i\leq n}, \|V_i - U_i''\| \leq \epsilon_i$ where $U_i'' = T'(U_i)$. By Definition 11, Q is matched with P.

Only if : By Definition 11, if Q is matched with P then $\exists T = (t_x, t_y, s_x, s_y, 0)$ such that $\forall_{1 \leq i \leq n}, [a_i - (s_x c_i + t_x)]^2 + [b_i - (s_y d_i + t_y)]^2 \leq \epsilon_i^2$. As shown in Proof 1, the n Circular Error Bounds $C_1, C_2, \ldots, C_n$ have common intersection. Thus, we have $\forall_{1 \leq i,j \leq n}$

$$\sqrt{[(a_i - s_x c_i) - (a_j - s_x c_j)]^2 + [(b_i - s_y d_i) - (b_j - s_y d_j)]^2} \leq \epsilon_i + \epsilon_j \qquad (6.7)$$

Re-arranging Equation 6.7, we have Equation 6.6. Thus, $\exists(s_x, s_y)$ such that $\forall_{1 \leq i,j \leq n}$, Equation 6.6 holds. Therefore, if $\mathbb{E}_{ij}$ defines an ellipse on the s_x-s_y plane, then $\exists(s_x, s_y)$ simultaneously resides in $\mathbb{E}_{i,j}$ for $1 \leq i, j \leq n$. That is, $\forall_{1 \leq i,j \leq n} \mathbb{E}_{ij}$ have common intersection.

6.2 Minimum Circular Error Bound

The polygon matching techniques presented in Section 6.1 only deal with matching queries subject to some tolerances (the tolerance vector E) and restrictions on transformation. By extending the idea of these techniques, we propose a similarity measure of polygons named Minimum Circular Error Bound (MCEB). Since MCEB is defined as the optimal value over all possible translations, it is a translation invariant similarity measure of polygons.

Definition 13 The Minimum Circular Error Bound ξ, where $\xi \in \mathbb{R}$, of a polygon $Q = \{U_1, U_2, \ldots, U_n\}$ comparing to another polygon $P = \{V_1, V_2, \ldots, V_n\}$ is defined as

$$\xi = \min_{\forall_{t_x, t_y} T = (t_x, t_y)} \max_{1 \leq i \leq n} \|V_i - T(U_i)\|$$

ξ can be calculated as follows. Let $V_i = (a_i, b_i)$ and $U_i = (c_i, d_i)$. Further assume that the tolerance vector $E = \langle \epsilon_1, \epsilon_2, \ldots, \epsilon_n \rangle$ where $\epsilon_1 = \epsilon_2 = \cdots = \epsilon_n$. The Circular Error Bound C_i is a circle with ϵ_i as its radius and $(a_i - c_i, b_i - d_i)$ as its center. If two Circular Error Bounds C_i and C_j intersect each other, we have

$$\sqrt{[(a_i - c_i) - (a_j - c_j)]^2 + [(b_i - d_i) - (b_j - d_j)]^2} \leq \epsilon_i + \epsilon_j \qquad (6.8)$$

Since $\epsilon_i = \epsilon_j$, we denote the value of ϵ_i and ϵ_j as ϵ_{ij}. The minimal value of ϵ_{ij} that Equation 6.8 holds is

$$\epsilon_{ij} = \frac{1}{2}\sqrt{[(a_i - c_i) - (a_j - c_j)]^2 + [(b_i - d_i) - (b_j - d_j)]^2}$$

The MCEB of the two polygons Q and P is

$$\xi = \max_{1 \le i,j \le n} \epsilon_{ij}$$

such that for $\epsilon_1 = \epsilon_2 = \cdots = \epsilon_n \ge \xi$, $\forall_{1 \le i,j \le n}$ C_i and C_j intersect each other. That is, for $\epsilon_1 = \epsilon_2 = \cdots = \epsilon_n \ge \xi$, Circular Error Bounds $C_1, C_2, \ldots, C_n$ of Q and P have common intersection and Q is matched with P under Proposition 1.

6.3 Characteristics

MCEB method is translation invariant but its sensitive to scaling and orientation so scale normalization and orientation standardization are required. The storage complexity is $O(n)$ and the computational complexity is $O(n^2)$. This method is not indexable.

Chapter 7

Experimental Results

Several experiments have been carried out. We will describe how we conduct the experiments and discuss the experimental results.

We compare the running time of the two-stage framework with the KD-Tree indexed NCS method using databases with different size and polygons with different number of vertices. We also compare the relative running time of the Hausdorff Distance method, the MRAM method, the MCEB method, and the NCS method without indexing. Besides running time, we also compare the quality of the visual ranking produced by the four methods.

7.1 Setup

We will now describe the environment and the programs that we constructed for the experiments as well as how the testing data are generated. An experiment is consisted of three steps as described follows.

1. A polygon data set is generated using one of the two polygon generators described in Section 7.1.1.

2. Several database construction programs are applied to the polygon data generated in step 1 to extract information need for query handling. These programs are described in Section 7.1.2.

3. A program is created for each polygon similarity measuring technique which allows user to select a target polygon from a list of templates and then carries out matching queries or similar queries. These programs are described in Section 7.1.3.

7.1.1 Polygon generation

We create two polygon generators for generating polygon data to be used as testing data. Each polygon is generated using Algorithm 7.1.

The two polygon generators differ in the distribution of number of polygons in equivalent classes. The first generator does not assume any distribution of the number of polygons in the equivalent classes. The second generator assumes that the distribution is uniform. Both polygon generators take two parameters. One is the number of vertices of the polygons to be generated. Another one is the total number of polygons to be generated.

Algorithm 7.1 Polygon Generation

/* *Input: n is the number of vertices of the polygon to be generated.* */

/* *Simple() is used to check whether $x[]$ and $y[]$ represent a non-crossing polygon* */

/* *StandardizeDirection() is used to re-organize $x[]$ and $y[]$ such that vertices are placed clockwise.* */

/* *random() returns a random value in the range* $[0, 1]$. */

$finish$ = **false**

while (**not** $finish$)

  for $i = 1$ **to** n **do**

    $x[i] = random()$

    $y[i] = random()$

  end for

  if $Simple(x[], y[])$ **then**

    $StanardizeDirection(x[], y[])$

$finish$ = **true**

end if

end while

return $x[], y[]$ as the ordered coordinate list of the generated polygon

The $Simple()$ function in Algorithm 7.1 use a brute force method to determine whether the given ordered coordinate list contains crossing edges. The function will pick up all possible edge pair and check if the two edges cross each other. Algorithm 7.2 gives the outline of the $Simple()$ function.

Algorithm 7.2 Determining whether a given ordered coordinate list contain crossing edges

/* *Input:* $x[]$ *and* $y[]$ *contain the ordered coordinate list.* n *is the number of vertices of the given polygon.* */

/* $cross(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$ *determines whether* $\overline{AB}$ *and* $\overline{CD}$ *cross each other, where* $A = \langle x_1, y_1 \rangle$, $B = \langle x_2, y_2 \rangle$, $C = \langle x_3, y_3 \rangle$, *and* $D = \langle x_4, y_4 \rangle$. */

for $i = 1$ **to** $n - 2$ **do**

  for $j = i + 1$ **to** $n - 1$ **do**

    if $cross(x[i], y[i], x[i+1], y[i+1], x[j], y[j], x[j+1], y[j+1])$ **then**

      return false

    end if

  end for

end for

return true

7.1.2 Database construction

We have created several programs to generate database files required for the query handling programs described in Section 7.1.3.

Polygon normalizer

We create a program which takes the output of the polygon generators as input and standardizes the orientation of the polygons in such a way that the longest edge of the polygon is aligned with the y-axis. If there are more than one edges having the longest length, the program simply picks the first one it encountered. After the polygon is re-oriented, it is then scaled to have a unit bounding box. Figure 7.1 illustrates the idea of polygon orientation standardization and scale normalization.

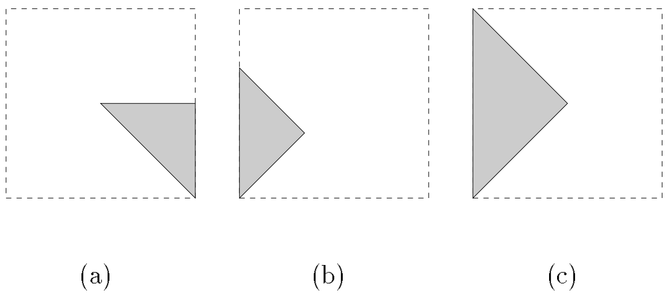


Figure 7.1: Orientation standardization and scale normalization of polygons
The dash line box is an unit bounding box. (a) the original polygon. (b) the polygon after orientation standardization. (c) the polygon after scale normalization.

SBSD database builder

This program takes the normalized polygon data and computes the SBSD of the polygons using the definition of Section 3.2.

MRAI database builder

This program takes the normalized polygon data and computes the MRAI of the polygons using Algorithm 5.1. The frame buffer used is 64 pixels by 64 pixels large and MRAI are recorded up to resolution level 3.

NCS KD-Tree indexing builder

This program takes the normalized polygon data and construct the KD-Tree for NCS. We use 4 vertices as the key, meaning that the key is a 8 dimensional key. For a n-gon, n entries, corresponding to the n possible 4 vertices sequences, are inserted to the index tree. The whole index tree is saved for future queries.

7.1.3 Query processing

We have created several programs to handle shape queries using the two-stage approach.

Template selector

This program extracts the first hundred of polygon out of a polygon database and let the user select one of them as the target polygon of a shape query.

SBSD filter

This program takes the output of the template selector, computes the SBSD of the target polygon, and then selects all model polygons inside the database that within the user specified Hamming Distance tolerance from the SBSD of the target polygon.

Matching query handler

A matching query handler is created for each of the aforesaid methods. These marching query handlers take a tolerance as input and then find out the set of model polygons, from those selected by the SBSD filter, that matched the target polygon subject to the tolerance. These programs load all the data they need, e.g. MRAI, KD-Tree, etc., into memory during operation.

Similar query handler

A similar query handler is created for each of the aforesaid methods. These similar query handlers take a integer number, m, as input and then find the m model polygons, from those selected by the SBSD filter, that are most similar to the target polygon. The output polygons are ranked with respect to their similarity to the target polygon. These programs also load the data they need into memory during operation.

Result display program

This program takes the output of any matching query handler or similar query handler and display the polygons.

7.2 Running time comparison

In this session, we compare the running time of the two-stage framework with the NCS approach using KD-Tree indexing. We also compare the running time of different polygon similarity measuring techniques under the two-stage framework. All the experiment are conducted on a UltraSparc 1/140 workstation.

7.2.1 Experiment I

In this experiment, we test the running time of the two-stage framework with the KD-Tree indexed NCS method. Three different methods, namely the Hausdorff Distance method, the MRAM method, and the MCEB method are implemented using the two-stage framework. Five polygon databases are used in this experiment. Each of them contains 9000 polygons with the same number of sides. The five databases contain polygons of 4 to 8 sides respectively. The distribution of the number of polygons inside equivalent classes is assumed to be uniform. Table 7.1 summaries the information of these databases.

The average matching query processing time of the four methods on these databases are shown in Table 7.2. The fastest method in handling each of databases are boxed. A chart corresponding to Table 7.2 is shown in Figure 7.2.

We observe that the NCS method with KD-Tree indexing is the fastest method when the polygons being handled are with small number of sides, e.g. 4 sided as shown in experiment I. When the number of sides of polygons increases, the three methods using

Table 7.1: Polygon databases for running time experiment I

n-gon	**Equivalent classes**	**Polygons per class**	**Total**
4	2	4500	9000
5	4	2250	9000
6	9	1000	9000
7	15	600	9000
8	30	300	9000

Table 7.2: Average running time of experiment I

n-gon	**MRAM**	**MCEB**	**Hausdorff Distance**	**NCS with KD-Tree**
4	0.0833 sec	0.1633 sec	0.3833 sec	$\boxed{0.0333 \text{ sec}}$
5	$\boxed{0.0500 \text{ sec}}$	0.1000 sec	0.2333 sec	0.0833 sec
6	$\boxed{0.0333 \text{ sec}}$	0.0500 sec	0.1167 sec	0.1167 sec
7	$\boxed{0.0167 \text{ sec}}$	0.0333 sec	0.0833 sec	0.1500 sec
8	$\boxed{0.0167 \text{ sec}}$	$\boxed{0.0167 \text{ sec}}$	0.0500 sec	0.2000 sec

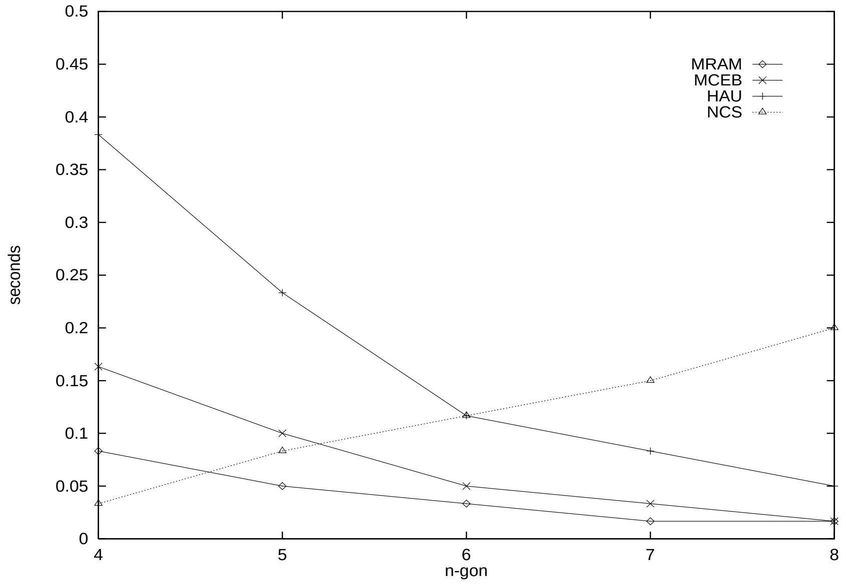


Figure 7.2: Average running time of experiment I

the two-stage framework are relatively faster than the NCS method. Using experiment I as example, the MRAM method become the fastest when the sides of polygons is larger than or equal to 5. When the number of sides increases to 6 or more, all the method using the two-stage framework are faster than the NCS method.

This is the expected result and can be explained as follows. For a polygon database with m n-sided polygons, mn entries will be inserted into the system using NCS method with KD-Tree indexing, as described in Section 2.5. For the two-stage framework, m entries will be inserted into the database no matter how many sides the polygons have. When handling shape queries, the NCS method has to consider all the mn entries which are organized in an index tree. However, the two-stage framework approach only have to consider $\frac{m}{E}$ entries where E is the number of equivalent classes and the distribution of polygons in equivalent classes is assumed to be uniform. As the number of sides of polygons, n, increases, the NCS method have more entries to consider while the two-stage framework approach has less, since E increases with n.

7.2.2 Experiment II

In this experiment, we test the running time of the four methods on databases with different size. 5 databases are used in this experiment which contain 7200, 9000, 10800, 12600, 14400 6-sided polygons respectively. The average matching query processing time of the four methods on these databases are shown in Table 7.3. The fastest method in handling each of databases are boxed. A chart corresponding to Table 7.3 is shown in Figure 7.3.

Table 7.3: Average running time of experiment II

Total number of polygons	**MRAM**	**MCEB**	**Hausdorff Distance**	**NCS with KD-Tree**
7200	$\boxed{0.0333 \text{ sec}}$	0.0333 sec	0.1000 sec	0.0500 sec
9000	$\boxed{0.0333 \text{ sec}}$	0.0500 sec	0.1167 sec	0.1167 sec
10800	$\boxed{0.0333 \text{ sec}}$	0.0500 sec	0.1500 sec	0.1333 sec
12600	$\boxed{0.0500 \text{ sec}}$	0.0667 sec	0.1667 sec	0.1500 sec
14400	$\boxed{0.0500 \text{ sec}}$	0.0833 sec	0.2000 sec	0.1833 sec

We observe that the running time of all the methods increase as the number of polygons in databases increase. It is because the number of database entries needed to be processed increase as the total number of database entries increase for both the two-stage framework and the KD-Tree indexed NCS method. However the percentage of increase in running time for the two approaches is different. Table 7.4 shows a normalized version of the result in Table 7.3. The results of each method is normalized using the running time of that method on the database with 7200 polygons.

We found that the rate of increase of the KD-Tree indexed NCS method is larger than those of the three methods using the two-stage framework approach. For example, when the number of database entries doubled from 7200 to 14400, the running time of the KD-Tree indexed NCS method is more than 300% of that on the database with 7200

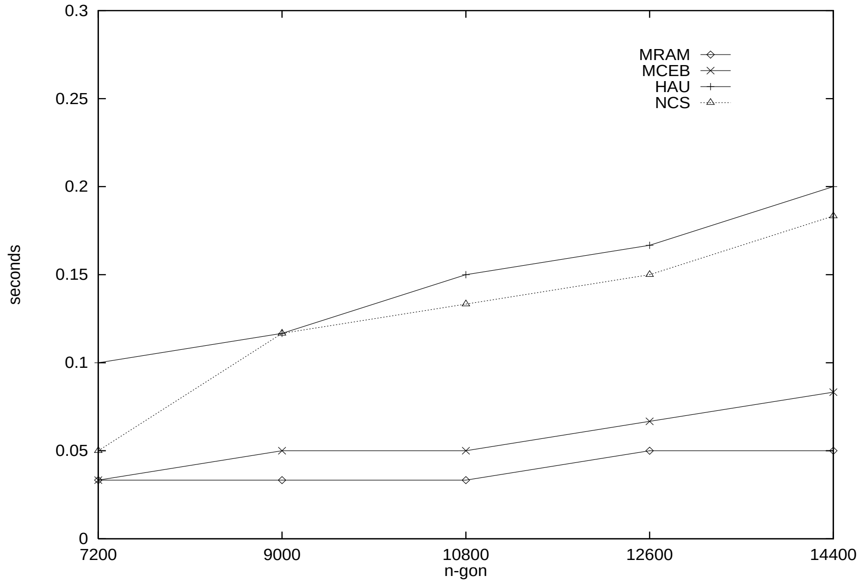


Figure 7.3: Average running time of experiment II

Table 7.4: Normalized average running time of experiment II

Total number of polygons	MRAM	MCEB	Hausdorff Distance	NCS with KD-Tree
7200	1.00	1.00	1.00	1.00
9000	1.00	1.50	1.17	2.33
10800	1.00	1.50	1.50	2.67
12600	1.50	2.00	1.67	3.00
14400	1.50	2.50	2.00	3.67

entries. Yet, the running time of other three methods are less than 300% of their running time on the database with 7200.

This result can be explained as follows. When a n-gon is added into a database, n entries will be inserted to the KD-Tree index in the case of the NCS method. However, only one entry will be inserted to the database used by those two-stage framework methods. Assuming an uniform distribution on the number of polygons in equivalent classes, the probability that this newly inserted entry have to be handled by the second stage polygon similarity measuring technique is $1/E(n)$, where $E(n)$ is the number of equivalent classes of n-gons. Thus, the effective increase in the number of database entries to be processed per polygon added is n for the KD-Tree indexed NCS method, and $1/E(n)$ for methods using the two-stage framework.

7.2.3 Experiment III

In this experiment, the running time of handling similar query is compared. The databases used are the same as in Section 7.2.1. However, the NCS method used in this experiment is different from the one in Section 7.2.1. Since the NCS method with KD-Tree indexing cannot handle similar queries, we modify it such that no indexing is used. We use the polygon similarity measuring technique of the NCS method, i.e. measuring polygon similarity by Euclidean Distance of vertex coordinates, in the second stage of the two stage framework. The average similar query processing time of the four methods on the databases are shown in Table 7.5. The fastest method in handling each of databases are boxed.

The result obtained in this experiment agrees with the computational complexity of the four methods described in Section 2. Recall that the computational complexity of the MRAM method, the MCEB method, the Hausdorff Distance method, and the NCS method are $O(L)$, $O(n)$, $O(n^2)$, and $O(n^2)$ respectively where L is the size of the MRAI used in the MRAM method. Thus, the NCS method successes in being one of the fastest method in the experiment. Though the MCEB method and the Hausdorff

Table 7.5: Average running time of experiment III

n-gon	**MRAM**	**MCEB**	**Hausdorff Distance**	**NCS**
4	0.1833 sec	0.1833 sec	0.4167 sec	$\boxed{0.1667 \text{ sec}}$
5	$\boxed{0.0833 \text{ sec}}$	0.1000 sec	0.2500 sec	$\boxed{0.0833 \text{ sec}}$
6	$\boxed{0.0500 \text{ sec}}$	$\boxed{0.0500 \text{ sec}}$	0.1167 sec	$\boxed{0.0500 \text{ sec}}$
7	$\boxed{0.0333 \text{ sec}}$	$\boxed{0.0333 \text{ sec}}$	0.1000 sec	$\boxed{0.0333 \text{ sec}}$
8	$\boxed{0.0167 \text{ sec}}$	$\boxed{0.0167 \text{ sec}}$	0.0500 sec	$\boxed{0.0167 \text{ sec}}$

Distance method have the same computational complexity, the MCEB method has a smaller constant factor so it is relatively faster than the Hausdorff Distance method. In general, L is far larger than n. However, since the MRAM method uses a multi-resolution comparing strategy with early rejection, it is also one of the fastest method in the experiment. When the number of sides of polygons increases, the four methods tend to have the same running time. It is because in these cases, the number of equivalent classes is relatively large and the number of entries to be compared in a shape query is relatively small. Thus, the running time is dominated by the setup cost of the programs rather the comparison cost of different methods.

7.3 Visual ranking comparison

Three experiments are carried out to compare the visual ranking of polygons produced by different methods. We will now describe these experiments and their results.

7.3.1 Experiment I

In this experiment, 50 polygons are generated (Figure 7.4). Given the first and the last polygon, 48 intermediate polygons are generated using linear interpolation. This method gives us a set of polygons as well as an ordering according to their similarity to the first polygon. For each similarity measuring technique, we perform a similar query using the first polygon as the target polygon. We compare the visual ranking produced by each method with the original ordering. The number of polygon mis-ranked is used as the quality measure of a visual ranking where a small number indicate a good quality visual ranking.

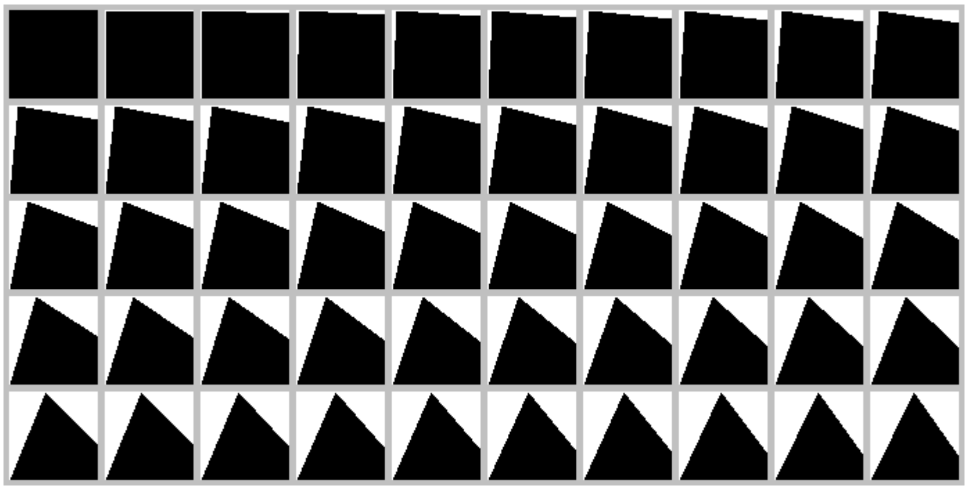


Figure 7.4: Polygon data for visual ranking experiment I

Polygons are in black color and is arranged top to bottom, left to right. The first one and the last one are used to generate the 48 intermediate polygons by linear interpolation.

From experiment, we find that all the four methods produce visual rankings exactly as the original ordering.

7.3.2 Experiment II

In this experiment, 50 polygons are generated using the same interpolation technique as described in Section 7.3.1, but with different pair of polygons for interpolation process. The 50 polygons are shown in Figure 7.5. The visual rankings produced by the four methods are also compared as described in Section 7.3.1.

From experiment, we find the Hausdorff Distance method produces a visual ranking

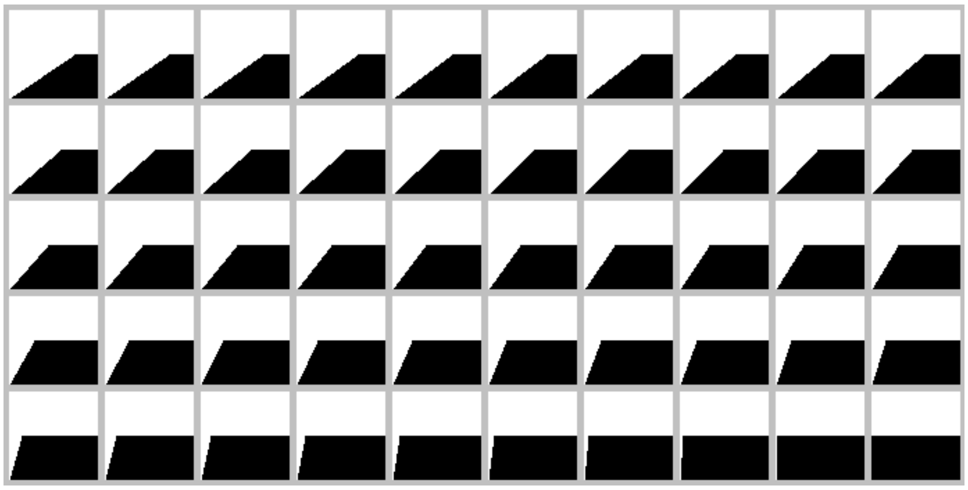


Figure 7.5: Polygon data for visual ranking experiment II

with a quality measure 13. The visual ranking produced by the Hausdorff Distance method is shown in Figure 7.6. All other three methods succeed in producing visual rankings exactly the same as the original ordering. Thus, the Hausdorff Distance method is not as good as others in handling this polygon data set.

7.3.3 Experiment III

In this experiment, 40 polygons (Figure 7.7) are generated using the same technique as the previous two experiments. The first 20 polygons are generated by interpolating the first and the 20th polygon. The last 20 polygons are generated by interpolating the 21th and the last polygon. Similar queries are performed using the four methods with the first polygon as the target polygon.

We find that the MRAM method and the NCS method fail to produce visual ranking the same as the original ordering while the MCEB method and the Hausdorff Distance method success in doing so. Both the MRAM method and the NCS method produce visual ranking with quality measure 19. The visual ranking produced by the MRAM method is shown in Figure 7.8 and the one produced by the NCS method is shown in Figure 7.9.

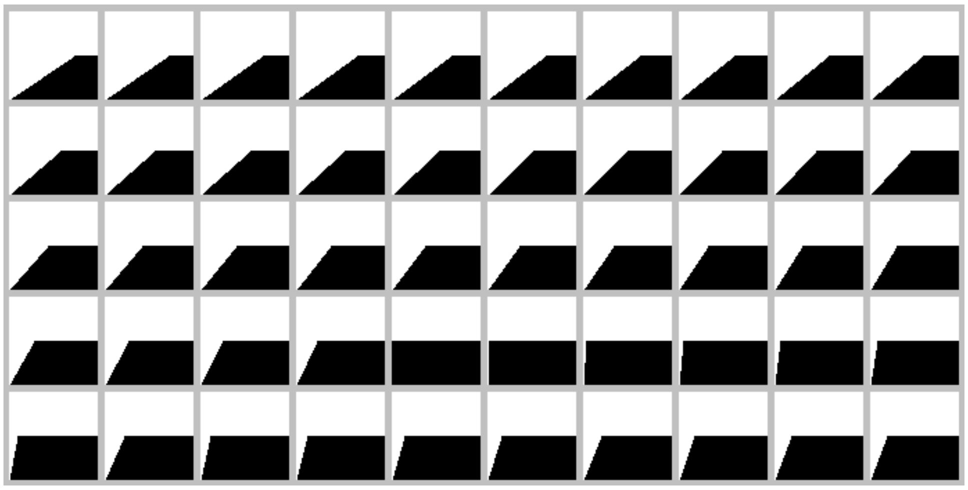


Figure 7.6: Visual ranking produced by the Hausdorff Distance method in visual ranking experiment II

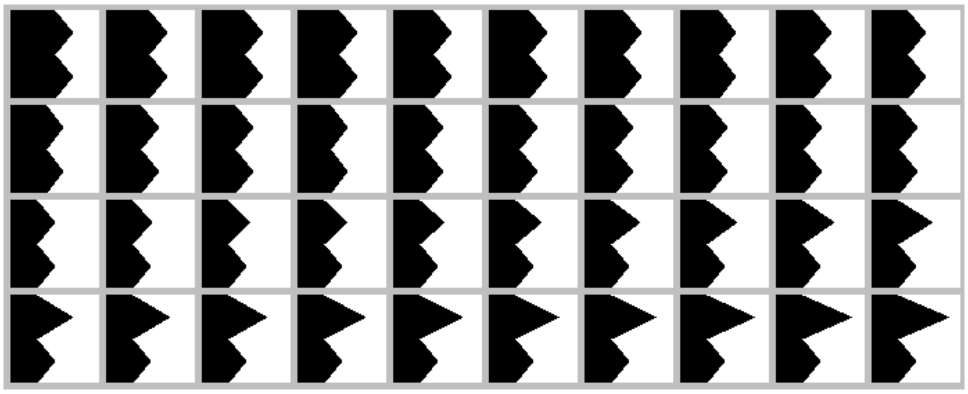


Figure 7.7: Polygon data for visual ranking experiment III

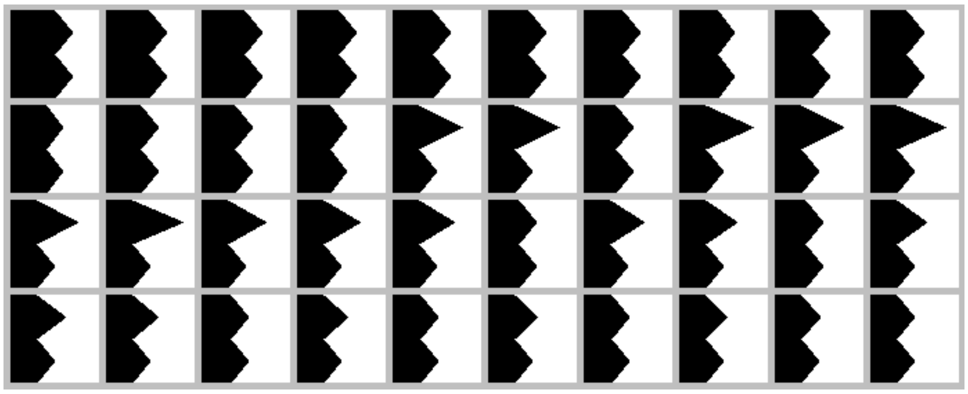


Figure 7.8: Visual ranking produced by the MRAM method in visual experiment III

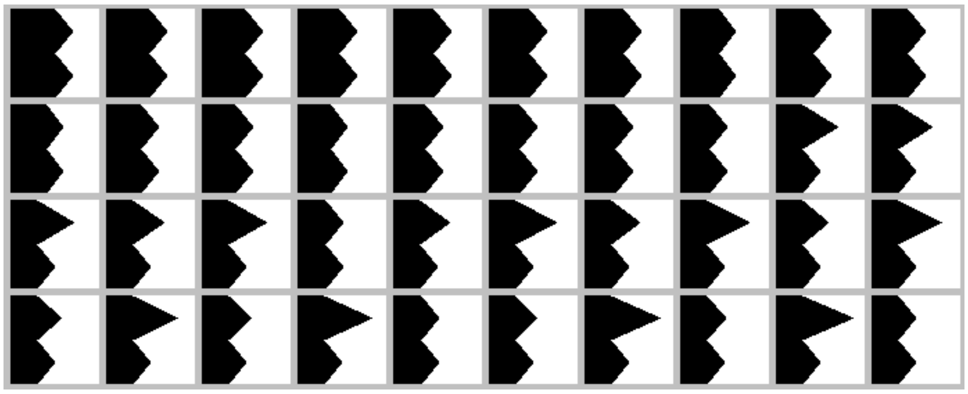


Figure 7.9: Visual ranking produced by the NCS method in visual experiment III

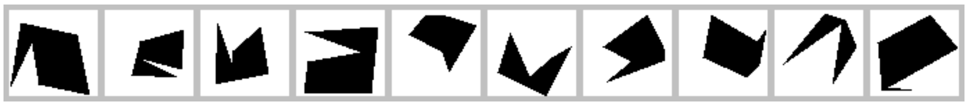
7.3.4 Conclusion on visual ranking experiments

From these three experiments, we find that the MCEB method is the only one that produces desirable visual ranking in all experiments. Other methods all fail to do so in either one of the experiments.

Figure 7.10 and Figure 7.11 show the results of similar queries produced by the four methods on the databases created for the running time experiments in Section 7.2. The queries are made using the first polygon as the target polygon and the 10 polygons that are most similar to the target polygon based on the four methods are retrieved.

(a) MCEB

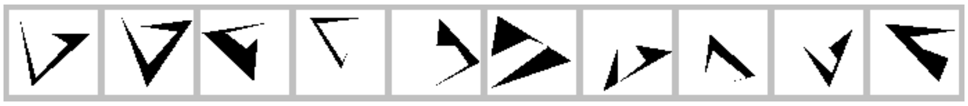
(b) NCS



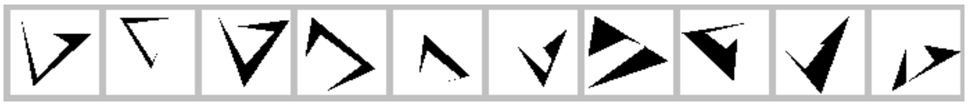
(c) Hausdorff Distance

(d) MRAM

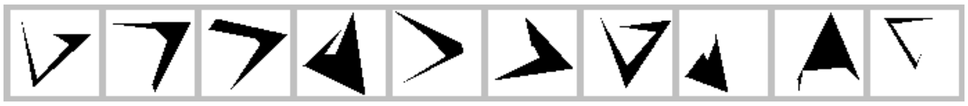
Figure 7.10: Similar query result I



(a) MCEB



(b) NCS



(c) Hausdorff Distance

(d) MRAM

Figure 7.11: Similar query result II

Chapter 8

Discussion

In this session, we will talk about the shortcomings of the two-stage framework and possible extensions to tackle them. We also discuss the relaxation of the restrictions of the two-stage framework.

8.1 N-ary Shape Descriptor

From Table 3.1, we observe that the number of equivalent classes are relatively small when the polygons being handled are with small number of sides. For example, all triangles will be in one equivalent class and all polygons with 4 sides will be partitioned into two equivalent classes. Therefore, SBSD may not be a good method for polygon classification in the two-stage approach in these situations since the pruning effect is not to good.

A possible solution to this problem is to record the angle of a vertex in more discrete levels, rather than convex and concave only. The idea is to generalize the BSD into N-ary Shape Descriptor (NSD) such that number of equivalent classes increases. NSD is defined as follow.

Definition 14 Let NSD^k denotes a NSD with k discrete levels of a polygon and Δ_i denotes the interior angle of the ith vertex of the polygon, then $\text{NSD}^k = a_1 a_2 \ldots a_{n-1} a_n$ where n is the number of vertices of the polygon and

$$a_i = j, j \in \{0, 1, \ldots, k-1\}, \frac{j\pi}{k} \leq \Delta_i < \frac{(j+1)\pi}{k}$$

Like SBSD, the Standardized N-ary Shape Descriptor (SNSD) of a NSD can be defined as follow.

Definition 15

$$\mathrm{SNSD}^k = \min_{1 \le j \le n} M(\mathrm{NSD}_j^k)$$

where

$$\mathrm{NSD}_j^k = a_j a_{j+1} \dots a_{n-1} a_n a_1 a_2 \dots a_{j-1}$$

and $M(\mathrm{NSD}_j^k)$ denotes the magnitude of NSD_j^k when it is regarded as a base-k integer.

For example, if NSD^4 is used, there will be 2 equivalent classes for triangles instead of 1 in the BSD case. If NSD^8 is used, there will be 6 equivalent classes for triangles. Figure 8.1 shows 6 triangles and Table 8.1 shows the SBSD, SNSD^4 and SNSD^8 of these triangles.

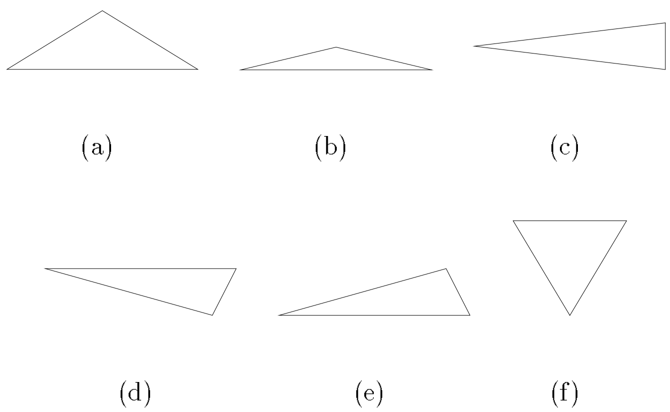


Figure 8.1: Triangles with different interior angles

8.2 Distribution of polygon equivalent classes

The distribution of number of polygons in equivalent classes is not studied in [6]. As the BSD techniques is used in our two-stage framework as a polygon classification method,

Table 8.1: SBSD, SNSD^4, and SNSD^8 of triangles

	SBSD	**NSD^4**	**NSD^8**
Figure 8.1(a)	000	001	002
Figure 8.1(b)	000	001	003
Figure 8.1(c)	000	000	011
Figure 8.1(d)	000	001	012
Figure 8.1(e)	000	001	021
Figure 8.1(f)	000	000	111

the distribution of equivalent classes will affect the performance of the framework since it will affect the pruning effect of the first stage of the framework.

From empirical experiments, we find that the distribution of equivalent classes is not uniform. The experiments are conducted as follows. Experiments on polygon with 4 sides, 5 sides, 6 sides, and 7 sides are carried out. In each experiment, 100000 simple degenerate closed polygons are generated using the algorithm described in Section 7.1.1. The SBSDs of the generated polygons are computed and the distribution of the SBSDs are recorded.

Table 8.2, Table 8.3, Table 8.4, and Table 8.5 shows the distributions of equivalent classes of the polygons with 4 sides, 5 sides, 6 sides, and 7 sides respectively. From the results, we find that the distribution varies with the number of sides of the polygon and it is not uniform. To have better pruning effect using BSD technique, we can split equivalent classes with large number of polygons into different classes. This can be done by using the NSD technique described in Section 8.1.

Table 8.2: Distribution of equivalent classes of 4-sided polygons

SBSD	0000	0001
Percentage	43.06%	56.94%

Table 8.3: Distribution of equivalent classes of 5-sided polygons

SBSD	00000	00001	00011	00101
Percentage	10.16%	77.61%	3.09%	9.14%

Table 8.4: Distribution of equivalent classes of 6-sided polygons

SBSD	000000	00001	000011	000101	000111	001001
Percentage	1.55%	49.33%	11.27%	21.84%	0.08%	14.43%
SBSD	001011	001101	010101			
Percentage	0.50%	0.50%	0.50%			

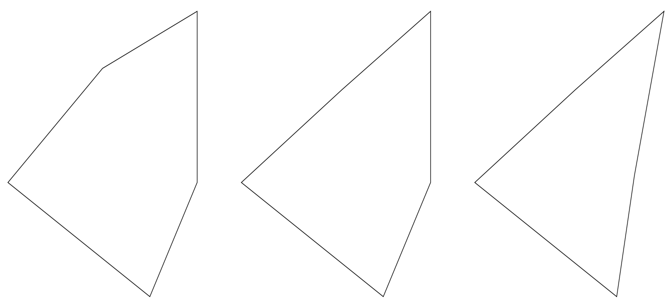
Table 8.5: Distribution of equivalent classes of 7-sided polygons

SBSD	0000000	0000001	0000011	0000101	0000111
Percentage	0.18%	18.61%	14.18%	21.26%	0.41%
SBSD	0001001	0001011	0001101	0001111	0010011
Percentage	30.25%	1.13%	1.10%	0.44%	1.92%
SBSD	0010101	0010111	0011011	0011101	0101011
Percentage	3.10%	1.16%	1.93%	1.19%	3.14%

8.3 Comparing polygons with different number of vertices

One of the problems of the two-stage framework is that only model polygons having the same number of sides as the target polygon will be selected for the second stage comparison. It is because the first stage of the framework select model polygons based on the Hamming Distance between their SBSDs and the SBSD of the target polygon. Since Hamming Distance only exist when the two input strings have the same length and two SBSDs have the same length only if their associated polygons have the same number of sides, only model polygons having the same number of sides as the target polygon can be selected in the first stage based on the Hamming Distance criteria.

A possible extension for the two-stage framework to tackle this problem is to reduce the vertices of the target polygon and initiate shape queries using this simplified target polygon in addition to the original one. The idea is to simplify the target polygon by making use of the curvature (turning angle) information of the vertices, i.e. remove the vertices with the smallest turning angle. Figure 8.2 illustrates the idea of polygon simplification using curvature information.



(a) (b) (c)

Figure 8.2: Polygon simplification using curvature information
(a) the original polygon (b) one vertex removed (c) two vertices removed

We suggest two schemes for specifying the simplification of target polygon:

1. User specifies the number of vertices to be removed. For example, the target polygon originally has n vertices and the user specify to remove 2 vertices, then the system should automatically initiate 3 shape queries using the original target polygon, the one with one vertex removed, and the one with two vertices removed. The advantage of this scheme is that user have a definite control over the degree of simplification.

2. User specifies a threshold value for the turning angle. When the system find that the turning angle of a vertex is less than the threshold, it automatically remove the vertex and initiate an additional shape query using the simplified target polygon. This process should be recursively applied to the simplified target polygon until no more vertex should be removed. The advantage of this scheme is that the simplification process will adjust itself according to the condition of the polygon to be processed.

The suggested extension only enable the comparison between the target polygon and model polygons with less number of vertices. With similar scheme, we can also make comparison between the target polygon and model polygons with more number of vertices possible. When a polygon is added to the database, this polygon then processed by either of the suggested simplification schemes. For each simplified polygon, as well as the original one, its corresponding data (SBSD, MRAI, KD-Tree index entries, etc) will be inserted into the database. This way, the target polygon can be effectively compared to model polygons with more number of vertices. One shortcoming of this scheme is that multiple database entries will be generated for a polygon.

8.4 Relaxation of assumptions

As stated in Chapter 1, our work assumes that the polygons being handled are simple non-degenerate closed polygons. However, the two-stage framework can handle polygons without these restrictions. The restrictions for the two-stage framework are actually the union of the restrictions, in a more strict sense, imposed by the techniques incorporated in both the first stage and the second stage of the framework.

Among the three assumptions, the requirement of polygon being non-degenerate is just a restriction introduced to reduce the complexity of polygonal shapes by removing vertices that are not important. The other two restrictions may or may not be relaxed depending on the techniques chosen. Table 8.6 gives a summary on the restrictions of the techniques we described in Chapter 2 and proposed in Chapter 5 and Chapter 6.

8.4.1 Non-degenerate

As mentioned, this restriction is introduced to reduce the complexity of polygons to be handled. This is good for techniques whose computational complexity depend on the number of vertices of polygons.

1. The BSD technique can work on degenerate polygons with the modification that the case of interior angle equal to π is classified either as convex or concave. Nevertheless, removing degenerate vertices will reduce the length of the BSD and thus reduce the computational requirement of the BSD technique. The same situation applies to the Potential-Based approach, the NCS method, the Hausdorff Distance method, the PCA method, and MCEB method.

2. Whether the polygons are degenerate or not is indifferent to the Freeman Chain Code method, the Moment method, the Rectangular Cover method, and the MRAM method. It is because the Freeman Chain Code method works on the boundary pixels and the other three methods work on area information of polygons. Thus, they can handle degenerate polygons as well as non-degenerate polygons.

8.4.2 Simple

1. Though the BSD technique is developed for simple polygons, it also works on complex polygons. However, the angles being recorded will not be interior angles but

the turning angle of next edge with respect to the previous one. Another result of applying BSD technique on complex polygons is that the analysis of number of equivalent classes will be invalid since the analysis is conducted base on simple polygons. However, the technique still works and can serve as a polygon classification method even a full study on the number of equivalent classes is not available.

2. The Chain Code approach requires the polygons being handled to be simple since there will be difficulty of determining the boundary to trace when the algorithm encounters a crossing of edges.

3. Whether the Moment approach and the MRAM method work on complex polygons or not depends on the scan-conversion technique incorporated in these two methods. If the scan-conversion technique used can handle complex polygons as well as simple polygons, then these two method can be used to handle complex polygons as well since both of them work on digitized shapes rendered in frame buffers.

4. The Potential-Based approach cannot handle complex polygons. It is true that its computation of artificial potential field only make use of the edges of polygons so the computation is valid even for crossing edges. However, this method has to place the template shapes "inside" the shapes to be classified. The "inside" of a complex polygon is hard to define since a complex polygon may have more than one closed regions. Thus, placing the template shape inside any closed region of a complex polygon will only limit its growth within that closed region but not the whole polygon.

5. The Rectangular Cover approach will be able to handle complex polygons if the rectangular cover generator can handle complex polygons as well as simple polygons.

6. The NCS method, the Hausdorff Distance method, the PCA method, and the MCEB method have no problem handling complex polygons since they all work on ordered point sets.

8.4.3 Closed

1. The Moment approach, the Rectangular Cover approach, and the MRAM method require the polygons being handle be closed because all these methods work on the area of the polygons and area is only computable when the polygons are closed.

2. The Potential-Based approach will not work on open polygon since it has to place the template shapes inside the polygons to be classified. This cannot be done on an open polygon.

3. Other methods have no problem on handling open polygons. The Freeman Chain Code approach only makes use of the boundary of the polygons and it still works even when the boundary is not closed. The MCEB method, the Hausdorff Distance method, the PCA method, and the NCS method can handle open polygons since all of them work on ordered coordinate lists which are also available even the polygons are open.

Table 8.6: Restrictions for different techniques

	Non-degenerate	**Simple**	**Close**
BSD	Preferable†	No	No
Freeman Chain Code	No	Yes	No
Moment	No	Depends‡	Yes
Rectangular Cover	No	Depends‡	Yes
Potential-Based	Preferable†	Yes	Yes
NCS	Preferable†	No	No
Hausdorff Distance	Preferable†	No	No
PCA	Preferable†	No	No
MRAM	No	Depends‡	Yes
MCEB	Preferable†	No	No

† Computational complexity reduced if the polygons are non-degenerate.

‡ Depends on the feature extraction technique used.

Chapter 9

Conclusion

We have proposed a two-stage framework for the polygon retrieval task which incorporates qualitative and quantitative measures of polygons in the first stage and second stage respectively:

1. The first stage uses SBSD as a mean to prune the search space and reduce the number of polygons needed to be compared with the target polygon.

2. The second stage incorporates any available polygon matching and similarity measuring technique to compare model polygons with the target polygon.

This two-stage framework is more efficient than model-driven approach since it reduces the number of polygons needed to be compared with the target polygon. It also avoids the difficulty and inefficiency of maintaining complex multi-dimensional index structures as data-driven approach methods do. Instead, it uses string as index which is well studied and efficient indexing techniques are available. The MRAM method, the MCEB method, and the Hausdorff Distance method are implemented using the two-stage framework and are compared with KD-Tree indexed NCS method. The experiments show that the three methods using two-stage framework are more efficient than the KD-Tree indexed NCS method when polygons to be handled have sufficiently large number of sides (5, in our experiments). The efficiency gain of using the two-stage framework approach increases as the database grows.

We also propose two polygon similarity measuring techniques named MRAM and MCEB. The MRAM method is an area-based technique incorporating multi-resolution Quadtree area coding. It compares polygons from coarse resolution to fine resolution and takes advantage of early rejection when dissimilarity is found in coarse resolution. Our experiments show that the MRAM method, implemented using the two-stage framework approach, is not only more efficient than the KD-Tree indexed NCS method but also the fastest one among the three methods using the two-stage framework approach.

The MCEB method measures the similarity between two polygons by a number derived from the distance between corresponding vertices of the two polygons. Our experiments shows that the MCEB method is less efficient than the MRAM method but more efficient than the Hausdorff Distance method and the NCS method. Our experiments also show that the MCEB produces visual ranking of polygons better than the MCEB methods, the Hausdorff Distance method, and the NCS method.

The contributions of our work include the proposal of the two-stage framework for polygon matching , the MRAM method, the polygon matching technique using CEB, and the translation invariant polygon similarity measure MCEB. We have also studied the characteristics of several existing methods and compare their performance with our methods.

There are some possible extensions to our two-stage framework. For example, we can enhance the BSD method or even use other polygon classification technique in the first stage of the framework in order to have a more uniform distribution on the number of polygons per equivalent class. Another extension is to enable the matching of polygons with different number of vertices under the two-stage framework.

Bibliography

[1] P. K. Agarwa, M. Sharir, and S. Toledo. Applications of Parametric Searching in Geometric Optimization. In *Proc. of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 72–82, 1992.

[2] H. Alt, B. Behrebds, and J. Blömer. Approximate Matching of Polygonal Shapes. *Annals of Mathematics and Artificial Intelligence*, 13:251–265, 1995.

[3] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An Efficiently Computable Metric for Comparing Polygonal Shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(3):209–216, 1991.

[4] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. Automatic and Semi-Automatic Methods for Image Annotation and Retrieval in QBIC. *SPIE*, 2420:24–35, 1995.

[5] B. Bhanu and O. D. Daugeras. Shape Matching of Two-Dimensional Objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(2):137–156, 1984.

[6] B. Bhavnagri. A Method for Representing Shape Based on an Equivalent Relation on Polygons. *Pattern Recognition*, 27(2):247–260, 1994.

[7] E. Bribiesca and A. Guzman. How to Describe Pure Form and How to Measure Differences in Shapes Using Shape Numbers. *Pattern Recognition*, 12:101–112, 1980.

[8] Y. Cheng, S. S. Iyebngar, and R. L. Kashyap. A New Method of Image Compression Using Irreducible Covers of Maximal Rectangles. *IEEE Trans. on Software Engineering*, 14(5):651–658, May 1988.

[9] J. H. Chuang. A Potential-Based Approach for Shape Matching and Recognition. *Pattern Recogntion*, 29(3):463–470, 1996.

[10] P. Cox, H. Maitre, M. Minoux, and C. Ribeiro. Optimal Matching of Convex Polygons. *Pattern Recognition Letters*, 9:327–334, 1989.

[11] L. S. Davis. Shape Matching Using Relaxation Techniques. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1(1):60–72, 1979.

[12] W. R. Franklin. Rays - New Representation for Polygons and Polyhedra. *Computer Vision, Graphics, and Image Processing*, 22:327–338, 1983.

[13] D. S. Franzblau. Performance Guarantees on a Sweep-Line Heuristic for Covering Rectilinear Polygons with Rectangles. *SAIM Journal on Discrete Mathemathics*, 2(3):307–321, 1989.

[14] H. Freeman. Boundary Encoding and Processing. In *Picture Processing and Psychopictorics*, pages 241–266.

[15] H. Freeman. Analysis of the Precision of Generalized Chain Codes for the Representation of Planar Curves. *Pattern Analysis and Machine Intelligence*, 3(5):533–539, Sept. 1981.

[16] W. I. Grosky and P. Mehrotra. Index-Based Object Recognition in Pictorial Data Management. *Computer Vision, Graphics, and Image Processing*, 52:416–436, 1990.

[17] L. Gupta and M. D. Srinath. Invariant Planar Shape Recognition using Dynamic Alignment. *Pattern Recognition*, 21(3):235–239, 1988.

[18] L. H., L. M. A., and L. M. R. J. Fast Hough Transform: A Hierarchical Approach. *Computer Vision, Graph, and Image Processing*, 36:139–161, 1986.

[19] B. Holt and L. Hartwick. Retrieving Art Images by Content: the UC Davis QBIC Project. In *Aslib Proceedings*, volume 46, pages 243–248, 1994.

[20] D. P. Huttenlocher and W. J. Rucklidge. A Multi-Resolution Technique for Comparing Images Using the Hausdorff Distance. Technical Report TR92-1328, Department of Computer Science, Cornell University, 1992.

[21] I. J. and K. J. The Adaptive Hough Transform. *IEEE Trans. PAMI*, 9(5):690–698, 1987.

[22] P. J., I. J., and K. J. A hierarchical approach to line extraction. *Proc. IEEE Computer Vision and Patt. Recogn. Conf., San Diego, CA*, pages 92–97, June 1989.

[23] H. V. Jagadish. A Retrieval Technique for Similar Shapes. In *Proc. of the ACM SIGMOD International Conference on the Management of Data*, pages 208–217, May 1991.

[24] I. King and T. K. Lau. A Feature-Bassed Image Retrieval Database for the Fashion, Textile, and Clothing Industry in Hong Kong. In *Proc. of International Symposium Multi-Technology Information Processing '96*, pages 233–240, 1996.

[25] J. G. Leu. Computing a Shape's Moments from its Boundary. *Pattern Recognition*, 24(10):949–957, 1991.

[26] C. C. Lu and J. G. Dunham. Shape Matching using Polygon Approximation and Dynamic Alignment. *Pattern Recognition Letters*, 14:945–949, 1993.

[27] M. Maes. Polygonal Shape Recognition Using String-Matching Techniques. *Pattern Recognition*, 24(5):433–440, 1991.

[28] R. Mehrotra and J. E. Gray. Feature-Based Retrieval of Similar Shapes. In *Proc. 9th International Conference on Data Engineering*, pages 108–115, 1993.

[29] R. Mehrotra and J. E. Gray. Similar-Shape Retrieval in Shape Data Management. In *IEEE Computer Magazine*, pages 57–62, Sept. 1995.

[30] E. E. Milios. Shape Matching Using Curvature Processes. *Computer Vision, Graphics and Image Processing*, 47:203–226, 1989.

[31] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker. The QBIC Project: Querying Images By Content Using Color, Texture, and Shape. *SPIE*, 1908:173–187, 1993.

[32] T. Pavlidis. A Review of Algorithms for Shape Analysis. *Computer Graphics and Image Processing*, 7:243–258, 1978.

[33] T. Pavlidis. Algorithms for Shape Analysis of Contours and Waveforms. *Proc. 4th Int. Jnt. Conf. on Pattern Recognition*, pages 70–85, 1978.

[34] T. Pavlidis. The Use of a Syntactic Shape Analyzer for Contour Matching. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 1:307–310, 1979.

[35] I. Pratt. Shape Representation Using Fourier Coefficients of the Sinusoidal. Technical Report ISSN 1361-6161, Department of Computer Science, Manchester University.

[36] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[37] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[38] T. W. Sze and Y. H. Yang. A Simple Contour Matching Algorithm. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 3(6):676–678, 1981.

[39] W. H. Tsai and S. S. Yu. Attributed String Matching with Merging for Shape Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):453–462, 1985.

[40] L. H. Tung and I. King. A Two-Stage Framework for Polygon Retrieval using Minimum Circular Error Bound [to be appeared in ICIAP'97].

[41] L. H. Tung, I. King, P. F. Fung, and W. S. Lee. A Two-Stage Framework for Efficient Simple Polygon Retrieval in Image Databases. In *Proc. of International Symposium Multi-Technology Information Processing '96*, 1996.

[42] L. H. Tung, I. King, P. F. Fung, and W. S. Lee. Two-Stage Polygon Representation for Efficient Shape Retrieval in Image Databases. In *Proc. of the 1st International Workshop on Image Databases and Multimedia Search*, pages 146–153, 1996.

[43] T. Wakahara. Shape Matching Using LAT and its Application to Handwritten Numeral Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(6):618–629, 1994.

[44] L. Xu, I. King, and S. Klasa. A PCA Approach on Attributed Graph Matching for Fast Pattern Classification [submitted].

[45] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough Transform (RHT). *Pattern Recognition Letters*, 11:331–338, 1990.

[46] C. T. Zahn and R. Z. Roskies. Fourier Descriptors for Plane Closed Curves. *IEEE Trans. on Computer*, c-21(3):269–281, 1972.