CARNEGIE MELLON UNIVERSITY

REPRESENTATIONS, FEATURE EXTRACTION, MATCHING AND RELEVANCE FEEDBACK FOR SKETCH RETRIEVAL

A DISSERTATION
SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
for the degree

DOCTOR OF PHILOSOPHY
in
ELECTRICAL AND COMPUTER ENGINEERING

by

Howard Wing Ho Leung

Pittsburgh, Pennsylvania
June, 2003

Abstract

Sketching is a natural way of input that provides an effective means of illustration. A sketch consists of multiple strokes that can be captured by pen-based devices to be stored in a database for future retrieval. Linear browsing is not feasible when the number of sketches in the database becomes large. Not only do we need a tool in retrieving sketches, but also we need an efficient system that can provide retrieval result with high recall and high precision. In this thesis, we propose a novel approach that includes several aspects in improving the sketch retrieval performance. Given a sketch, we propose to have *multiform representations* of this sketch in order to find at least one consistent representation when it is compared against other similar sketches under user variations. We then propose to perform *coarse-to-fine feature extraction* in order to capture the characteristics of the sketch at various levels. We build a classifier in shade region detection for hand-drawn sketches and for images. When two sketches are compared, we propose to have *global and local matching* that computes the similarity not only based on the shape information, but also based on other criteria such as spatial relations and the structures. In addition, we propose to extend traditional single component relevance feedback to *multiple component relevance feedback* in order to refine the retrieval result based on the user feedback. Finally, we will show our approaches for solving the partial matching problem. In the last part of this thesis we will show several prototypes that we have been implementing in order to demonstrate how sketch retrieval can be applied in real applications.

Acknowledgements

I would like to start by thanking my advisor Prof. Tsuhan Chen who has been so supportive during my Master and my PhD programs. He provides me with good guidance and leads me to explore different research directions. Under his supervision, not only do I learn how to solve problems, but also learn how to solve them in an efficient manner. In addition, he also has provided me a lot of feedback to improve my presentation skills. Throughout our group and individual meetings, he teaches us how to communicate effectively so that we can become good teachers ourselves. I have learnt how to become a successful researcher and I hope to carry on to make contributions to our community. I am very fortunate to become his student and join his group.

I would also like to thank Prof. B. V. K. Vijaya Kumar and Prof. Christos Faloutsos, Prof. S. K. Chang and Dr. Zon-Yin Shae for spending their valuable time as my thesis committee members. I would like to thank them for all the fruitful discussions that improve the quality of this work. It was a great experience to have been in the Electrical and Computer Engineering department at Carnegie Mellon University, in this atmosphere where I have the opportunity to meet many great faculties, staff and students. Besides, I would like to thank Dr. Belle Tseng for teaching me so many things during my summer interns.

My current and previous group mates Fu Jie Huang, Ta-Chien Lin, Deepak Turaga, Trista Chen, Xiaoming Liu, Cha Zhang, Claire Fang, Wende Zhang, Edward Lin, Sam Chen, Jessie Hsu, Simon Lucey, Jack Yu, Kate Shim, Avinash Baliga and Michael

Kaye broaden my views through the exchange of research ideas. I have developed the sense of belonging and I am proud to be in this AMP (Advanced Multimedia Processing) group. In addition to my group, I also enjoyed hanging out and doing fun activities with other ECE friends such as Pinky Pongbaipool, Poj Tangamchit, Wee-Seng Soh, Ece Guran etc...

I would like to give my special thanks to my distant friends Kit Shan Li and Jenny Li who have made me understand many things in life and have given me a lot of moral support. In addition, I also thank Wai-Yee Chan, Apple Cheung, Elsa Luu and Quan Tran for the encouragement especially when I was facing difficult moments. I cannot possibly list all my friends' names here but I would like to thank them all.

Finally, I would like to thank my family: my father, my mother and my brothers. Although they live in other places, they have always been there for me to provide me with support. I dedicate this thesis to my parents.

Table of Contents

List of Tables

List of Illustrations

1. Introduction

Pen computing has become more and more important in our society [40] due to the popularity of pen-based devices such as TabletPC [1] that recently came out. Pen-based devices provide users with a natural way of input for drawing sketches. A sketch can consist of handwritten notes, symbols, free-form hand-drawings, annotations on a document etc. It will be very useful to store the sketches in a database and then retrieve them later by providing a simple sketch query. For example, in a classroom, the teacher may write and draw the lecture notes on the whiteboard that can be captured and stored in a database. Later students can retrieve relevant lecture sketches from the database by drawing a sketch as the query.

Query by sketch falls into the category of content-based image retrieval (CBIR). QBIC [14] was the first CBIR system and it also supports query by sketch. Global features such as area, circularity, eccentricity, etc., are used in shape matching. Matusiak et al. [33] proposed another approach to sketch-based images database retrieval by using Curvature Scale Space (CSS) to match contours. In Sciascio and Mongiello's system [41], the Fourier descriptors are used for shape comparison and they use relevance feedback to improve the retrieval performance for content-based image retrieval over the web. All the above systems assume that the query consists of a single shape.

Lopresti et al. [29][30] reported their work on matching hand-drawn pictures that they call "pictograms". This approach has a drawback that it treats the same hand-drawings with different stroke orders as a poor match. In order to make the system less

sensitive to the stroke order, Lopresti and Tomkins [31][32] proposed to match the strings block by block. However, poor match may still result if a stroke is drawn in reverse direction (i.e., when the start point and the end point of a stroke interchange). Under these approaches, string matching is performed for the alignment based on the time sequence. They may work well for handwritings or pen gestures [28] when the strokes have certain sequence pattern but may not be suitable for unstructured free-form hand-drawings.

The Query by Visual Example (QVE) reported by Kato et al. [19] used correlation of the corresponding blocks between the edge maps for evaluating similarity. Due to the variations in drawing style, this correlation approach will hardly match two rough sketches. Del Bimbo and Pala [11] proposed to use elastic matching to retrieve images from the database based on the user sketch. However, this energy minimization technique may be too time consuming when it requires many iterations for the solution to converge.

In our prior work [23], we proposed a retrieval method for hand-drawn sketches. It is based on string matching by the alignment of the spatial order among the boundaries of the minimum bounding rectangles of the strokes in each of the x and y projections. In [24], we included the similarity in spatial relations between strokes in the computation of the overall similarity score. We have introduced another application of query by sketch in trademark retrieval [25].

The focus of this thesis is on finding an efficient sketch retrieval method to improve the retrieval performance in terms of archiving relevant materials from the database with minimum number of trials. Specifically, we target on developing novel

approaches in different aspects of the retrieval system. Given a sketch, we propose to create *multiform representations* of this sketch in order to find at least one consistent representation when it is compared against other similar sketches under user variations. We then propose to perform *coarse-to-fine feature extraction* in order to capture the characteristics of the sketch at various levels. When two sketches are compared, we propose to perform *global and local matching* that computes the similarity not only based on the shape information, but also based on other criteria such as spatial relations and the structures. Finally we propose to extend traditional single component relevance feedback to *multiple component relevance feedback* in order to refine the retrieval result based on the user feedback.

1.1. Multiform Representations

Different people may draw the same sketch in a different way due to variations in style. Moreover, sometimes even the same person may draw the same sketch differently due to user inconsistency. In some shape recognition approach, dominant points are detected from the contour in order to be used as feature points. On the other hand, in handwriting recognition, similar method has been proposed to split or merge the strokes in order to have a better representation for improving the recognition result. However, variations in general sketches are much more than a specific domain of handwritings therefore it is unlikely to find a good criterion for splitting and merging the strokes to form a single consistent representation. As a result, we propose to create multiform representations for each sketch. From the original representation of a sketch, we try to split the strokes into smaller stroke segments based on the dominant points to obtain the split representation. Then from the split representation, the stroke segments are merged if

they form a primitive shape to form the merged representation. We show that by combining all three representations the retrieval performance is better than each single representation.

1.2. Coarse-To-Fine Feature Extraction

We propose a to extract features in a coarse-to-fine manner. After preprocessing, the stroke hierarchy is constructed to capture the structural information of the sketch. The strokes are analyzed in order to detect for any shaded regions. We introduce a novel concept of *hyper-stroke* that is defined by a group of strokes inside a region enclosed by a boundary stroke. A hyper-stroke can be formed from strokes of a shaded region or it can also be derived from the stroke hierarchy. Coarser features representing those of a group of strokes are extracted from each hyper-stroke. On the other hand, finer features represented by primitive shape features are extracted from each basic stroke.

1.3. Global And Local Matching

We propose to compare the features at different levels in order to match them both globally and locally. While a sketch consists of multiple components and two sketches may have different number of components, it is necessary to first find a correspondence between the components in the two sketches. This correspondence is determined from global matching by minimizing the total cost (or maximizing the total similarity score) between components. Based on this correspondence, the similarity score between the components can be computed. On the other hand, local matching that compares attributes of only a few components independently can also be performed such as the matching of stroke hierarchy and the spatial relations.

1.4. Multiple Component Relevance Feedback

Traditionally, relevance feedback can be applied to update the query features of an object with a single component. However, multiple components exist in a sketch representation therefore extension is required in order to perform relevance feedback for updating query features of an object with multiple components. We also propose to update the weights for combining the matching results from the multiform representations in order to assign a higher weight for the better representation based on the user feedback.

This thesis is organized as follows. Chapter 2 describes the fundamentals of sketch retrieval. We will survey existing capture devices for sketches and the storage format. Then we provide a block diagram for the sketch retrieval system. Next we discuss our data collection and the experimental setup. Finally we explain the evaluation criteria for the retrieval performance.

Chapter 3 describes the multiform representation creation in our sketch retrieval system. We obtain the split representation by detecting the dominant points of the strokes and then obtain the merged representation by combining spited strokes that are likely to form one of the primitive shapes.

Chapter 4 introduces coarse-to-fine feature extraction, that tries to obtain semantic information in addition to low-level features. Novel ideas about capturing the structures of the sketches with stroke hierarchy and grouping a set of strokes into a hyper-stroke are discussed. We explain how classification is used to detect shade regions for both hand-drawn sketches and for images.

Chapter 5 includes the description of the global and local matching. The correspondence between multiple components is first determined and two approaches for solving this problem are analyzed. The similarity functions for various levels of features are provided to perform matching globally and locally.

Chapter 6 explains the multiple component relevance feedback. We introduce the traditional relevance feedback approach for object with one component and extend this approach to handle objects with multiple components. Moreover, we also describe our approach for updating the weight for combining similarity scores from multiform representations.

Chapter 7 discusses the partial matching problem. We introduce two matching schemes in order to find the correspondence between features. The first approach is dynamic programming based on the alignment of the spatially ordered strokes and the second approach is bistroke matching based on searching the best match for each pair of strokes according to both shape and spatial relation similarity.

Chapter 8 lists several applications based on our sketch retrieval system. We present the prototypes that we have been implementing.

We finally conclude with a summary of the contributions of the thesis and some future directions for research.

2. Fundamentals of Sketch Retrieval

This chapter provides some background materials for sketch retrieval. It outlines several important processes in handling sketches. It describes how sketches can be captured, how they can be stored and how they can be retrieved. We state the problems that we focus on solving along with the introduction of different stages in the retrieval process. The data collection process, the experimental setup and the retrieval performance evaluation criteria are also discussed.

In an office, a whiteboard can be used to sketch a plan, write down reminders or illustrate an idea to visitors during an informal discussion. Some video conferencing applications such as NetMeeting [34] incorporate an electronic shared whiteboard that facilitates information sharing among users. The shared whiteboard allows multiple users from different locations to do collaborative work. Traditionally, a whiteboard (or blackboard) is used in a classroom for the teacher to write down course materials for the students to learn. As a result, a shared whiteboard can be included in a multi-user virtual environment so that it is suitable for distance learning. In addition, more than one shared whiteboard can appear in a virtual environment and each whiteboard may be used for a specific function that can be associated with the surroundings. For example, in a virtual office building, users may use the whiteboard for checking important phone numbers or appointment times in their offices or they may leave a message for other users on the whiteboard in the corridor.

Nowadays many commercial products target for this purpose in providing a convenient way for users to draw sketches that can be interfaced with computers. Pen-based devices become more popular recently since it provides a nature way of input. We will list several technologies for capturing sketches and analyze their pros and cons.

After capturing the sketches, it is essential to store them in order to view them at a later time. The two most common storage formats for sketches are images and strokes. Their advantages and disadvantages will be discussed.

When the user needs to search for relevant sketches at a later time, a linear browsing of the data is not feasible if the number of items in the database is large. As a result, retrieval system is designed to solve this problem by providing the user a tool to search for relevant items in the database easily and efficiently. We will give the overview of a sketch retrieval system and associate with our proposed methods with the system.

This chapter is organized as follows. Section 2.1 provides a survey of input devices that can be used to capture sketches. Section 2.2 describes the storage format of sketches. Section 2.3 introduces a general sketch retrieval system and identify which parts of the retrieval problem we are attempting to solve. Section 2.4 introduces our data collection process and Section 2.5 describes the experimental setup. Finally Section 2.6 provides an explanation of the evaluation criteria for retrieval performance.

2.1. Capture Device for Sketches

In a classroom, blackboard and chalk have long been essential tools for teachers to convey the message to the students. In an office, the presence of a physical whiteboard and markers allows people to sketch their ideas. In these scenarios, people need to copy

down the writings or drawings on the whiteboard to their notebook in order to keep a record. To remove the burden of the manual copying, a whiteboard session can be captured electronically by using some capture devices at the same time as the content is created. The capture devices usually consist of two components: an input device and a visual feedback device. The input device allows the user to write and draw naturally and its underlying electronic components capture the pen-down sample points. The visual feedback device allows the user to see instantly what is on the board so that they can continue to write or draw consistently. There exists various kinds of capture devices to achieve this purpose and we will discuss them in the following sections.

2.1.1. Mouse and Monitor

The most common way of capturing pen strokes is to use a mouse and then display the captured strokes on the monitor. Since mouse and monitor are standard components of a computer, no extra hardware is required. The shared whiteboard application simply captures mouse coordinates when the button is pressed. The problem with this approach is that it is difficult to draw with a mouse because the user needs to keep his/her hand steady. As a result, drawing or writing with a mouse is slow and the resulting drawings may contain a lot of jittering.

2.1.2. Tablet and Monitor

The tablet is a pen-based device that can be attached to a computer and it allows users to draw naturally on it. While the user is drawing on the tablet, the visual feedback is provided on the monitor. But since the visual feedback area (monitor) is different from the drawing area (tablet), it can be difficult to control the pen to start at the specific

desired location after the user has a pause and then resumes drawing. Figure 1 illustrates an example of a tablet called graphire2 manufactured by WACOM [50].

Figure 1 Tablet manufactured by WACOM

2.1.3. Wireless Pen and Paper Pad

With this setup, the user can draw with a special pen on regular paper attached to a pad. This method is even more natural since the user is performing the same action as jotting notes on regular paper. However, the drawback in this approach is that if the user wants to erase something from the electronically captured content, he/she may need to choose special function from the application and the resulting change will not reflect on the regular paper. As a result, the content that is captured and the content that is actually drawn on the regular paper may be out-of-sync after the user does some editing. Figure 2 illustrates an example of this kind of commercial product called SmartPad manufactured by Seiko Instruments [42].

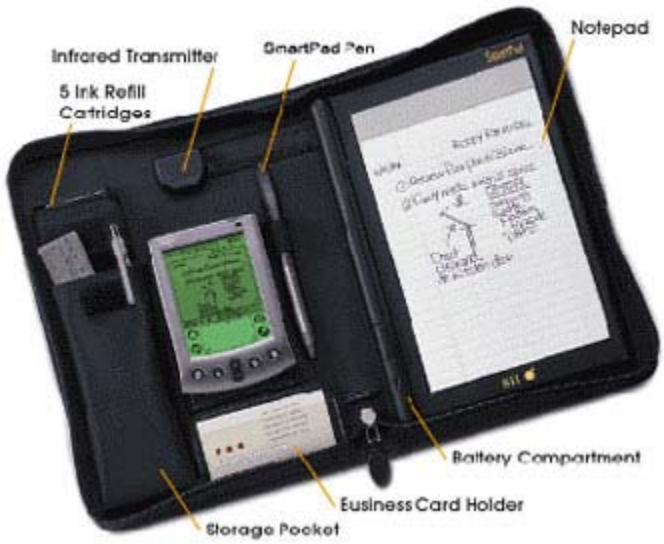


Figure 2 SmartPad manufactured by Seiko Instruments

2.1.4. Wireless Pen, Physical Whiteboard and Projector

In this setting, a capture bar is mounted on the side of a physical whiteboard to detect the coordinates of the wireless pen when the user uses it to draw or write on the whiteboard. The wireless pen is inkless meaning that it does not leave any physical mark on the whiteboard. Its tip has a switch and will be turned on while it is pressed towards the whiteboard during writing, thus sending the coordinates of the wireless pen to the capture bar. A projector is setup to project the resulting virtual ink back to the whiteboard to give the user instant visual feedback. Calibration is required at the beginning in order to map the cursor coordinates of the application to the physical location of the wireless pen such that the projection will overlap correctly with the designated drawing area of the whiteboard. Similar to the paper pad in the previous case, the physical whiteboard is served as both drawing area and the visual feedback area. The physical whiteboard provides a much bigger drawing area. However, this approach has the drawback that occlusion may occur because the projection may be blocked by the

user. There exist several commercial products for this kind of capture devices such as mimio [49] or eBeam [13]. An example setup with mimio and the projected whiteboard display is shown in Figure 3. A more detailed description about this particular setup used with our virtual environment can be found in [46].

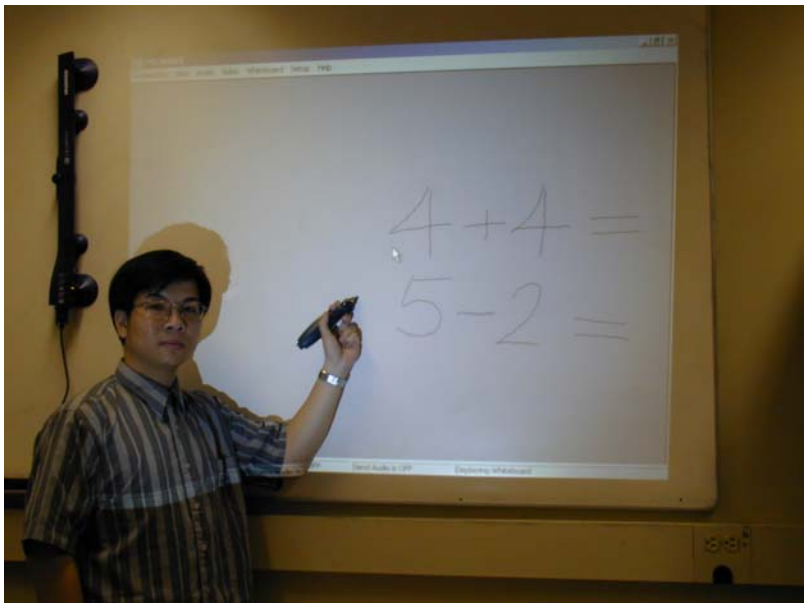


Figure 3 mimio and the projected whiteboard display

2.1.5. Stylus and Touch Screen

The user uses a stylus to write directly on top of a screen and whatever the user writes will be displayed instantly on the same screen. The screen is pressure sensitive in order to detect when and where the pen is down. The advantage of this setup is that the drawing area and the visual feedback area are integrated into the same device so that the inconsistency between drawing and displaying can be avoided. However, a touch screen is more expensive compared with other kinds of capture devices. On the other hand, as common to any typical new hardware, the cost is expected to decrease as time passes and this approach will be more and more common to the general public. The new product line Tablet PC that includes a pressure-sensitive tablet integrated into a laptop screen will

make shared whiteboard applications much more convenient to use. An example Tablet PC manufactured by Acer is shown in Figure 4 [1].

Figure 4 Tablet PC manufactured by Acer

2.2. Storage Format for Shared Whiteboard

The user may want to save the content of the whiteboard after being captured by the input device so that he/she may review the content at a later time. There are two common formats that the whiteboard content can be stored: as an image or as strokes.

2.2.1. Image

The content on the whiteboard can be saved as an image. The background whiteboard may be represented by white pixels and the drawing or writing can be represented by black pixels. Using this format, the memory required to store the whiteboard content as a raw image is fixed since the total number of pixels on the whiteboard is constant. The image may be compressed to reduce the space required for storage. The image represents only the final outcome and the intermediate results are not stored. Timing information is not included in the image format. One can save a whiteboard session as a video, i.e., save the whiteboard image buffer at different time instants, to show the progress over time. However, it will require a lot of storage space or

else the resolution will be poor if compressed with low bit rate by standard video compression techniques.

2.2.2. Strokes

An alternative solution to store the whiteboard content is to store them as strokes, i.e., save the sequence of the x and y coordinates captured from the input devices over time. Timing information is saved under this format therefore during playback the whiteboard session at any time instant can be easily displayed. Another advantage for the stroke format over the image format is the robustness to transformation. When an image is under rotation or scaling, the quality of the transformed image may be poor due to interpolation. However, since we are only transforming the stroke sample points while maintaining their connectivity, the resulting display can still have high resolution.

2.3. System Overview for Sketch Retrieval

The reason for storing the sketches is to be able to archive it at a later time. The relevant content needs to be archived according to what the user wants. Given a lot of captured sketches, a database can be formed and it will be useful if we are able to search through this database efficiently for relevant information according to our input query.

Retrieval based on sketches has several advantages:

1) Compared with keyword-based approach, retrieval based on sketches does not require the users to understand the context of the object. In keyword-based approach, each object first needs to be annotated which requires lots of manual labor. Moreover, different people may assign different keywords to the same sketch and sometimes it is difficult to describe something with words.

2) Compared with image retrieval, retrieval based on sketches contains more semantic information since each basic unit is a stroke instead of a pixel. This also means that retrieval based on sketches can be more robust to local variations.

3) Compared with traditional handwriting recognition, retrieval based on sketches provides a more flexible way of matching sketches. Handwriting recognition tries to map handwritten characters to a set of alphabets using the information that characters are drawn in certain order in time. On the other hand, a sketch is a more general representation therefore it is language independent and does not make assumption about the drawing order of the strokes.

A system overview of our sketch retrieval system is shown in Figure 5.

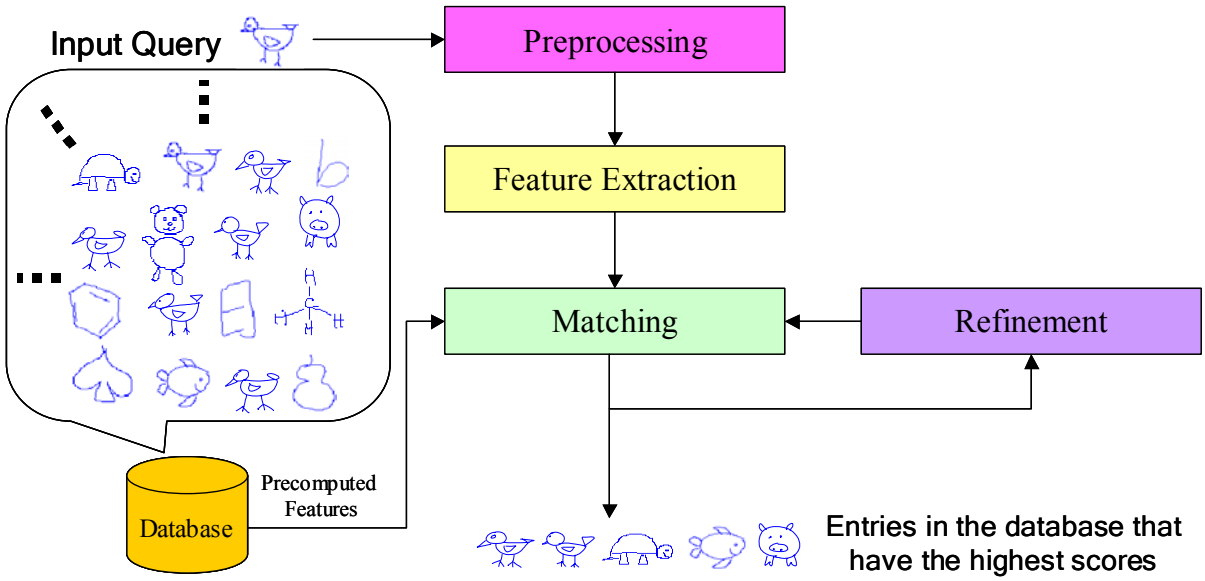


Figure 5 Block diagram of our sketch retrieval system

There are several stages in the sketch retrieval process: preprocessing stage, feature extraction stage, matching stage and refinement stage. In the preprocessing stage, image processing techniques are often applied in order to obtain a consistent representation of the data. In the feature extraction stage, features that can well describe

certain characteristics of the data are extracted. In the matching stage, the similarity scores between the query sketch and each sketch in the database are computed. The sketches in the database with the highest similarity scores are retrieved. In the refinement stage, the user has the option to provide relevance feedback to the system by indicating positive and negative examples. The system will retrieve new results after learning from those examples. Now we introduce specific problems at each stage that we focus on solving.

2.3.1. Preprocessing Stage

Raw data often contains useful information as well as unwanted noisy information due to different kinds of variations. The goal of the preprocessing stage is to remove some unwanted information in order to obtain a consistent representation of the data. In sketch retrieval, the variation of the sketch raw data may be due to user style such that the same sketch may be drawn in a different way by different people or the variation may be due to capturing process such that even the sketch drawn by the same person may appear differently at different times. Since there can be so many possibilities in terms of variations, finding a unique representation that is consistent over all kinds of variations is a difficult task, sometimes there may not be a solution. Therefore, instead of keeping only one representation, we propose to use multiform representations for each sketch such that it is more robust under different kinds of variations.

2.3.2. Feature Extraction Stage

After preprocessing, features that describe certain characteristics of the data are extracted. The choice of features is often based on heuristics and it depends on the data.

We propose a to extract features in a coarse-to-fine manner such that features at different levels are considered. As a result, in addition to low-level features, we also extract high-level semantic features from the representation after the preprocessing stage. For sketch retrieval purpose, coarse-to-fine feature extraction can be conceptually considered as looking at the sketch with different points of view. When coarse features are extracted, it can be considered as the case when the sketch is being viewed far away and when fine features are extracted, it can be considered as the case when the sketch is being viewed nearby. We propose several strategies in the coarse-to-fine feature extraction. Firstly the sketch is analyzed in order to detect for shaded regions since the strokes in a shaded region are better treated as a single unit. Then we construct the stroke hierarchy of each sketch to capture the structural information. We introduce a novel concept of *hyper-stroke* that is composed of a group of strokes inside a region enclosed with a boundary stroke. Shape features of basic strokes and spatial relations between strokes are also extracted.

2.3.3. Matching Stage

The goal of the matching stage is to compare the features of the query with the features of the items in the database and compute the similarity score between them. The similarity scores are sorted and those items in the database with high scores are retrieved. We propose to compare the features at different levels in order to match them both globally and locally. While a sketch consists of multiple components and two sketches may have different number of components, it is necessary to first find a correspondence between the components in the two sketches. Based on this correspondence, the similarity score between the matched components can be computed. In addition, spatial

relations between strokes are also considered in addition to shape features when the similarity is computed.

2.3.4. Refinement Stage

In the refinement stage, the user has the option to provide relevance feedback to the system by indicating positive and negative examples. The challenge is to figure out from the feedback examples what to update in order to increase the retrieval performance and to adapt better to what the user needs. Current approaches include 1) modifying the features of the query to make them closer to the positive examples and farther away from the negative examples; and 2) updating the weights according to how much impact they have on the feedback examples. We first propose to extend traditional query feature movement approach to handle objects with multiple components. We also propose to update the weights for combining the matching results from the multiform representations in order to assign a higher weight for the better representation based on the user feedback.

2.4. Data Collection

For the data collection, a PDA device (Compaq iPaq Pocket PC) is used to capture the sketches. There are sketches from 11 people in our database. During each session, each person draws 1 to 3 repetitions (depending on how fast they draw and how much time they can allocate for the session) for each of the 37 classes of sketches. As shown in Figure 6, the classes include Chinese characters, Korean characters, English words, mathematical equations, chemical structure, flow diagram and free-form hand-drawings such that they cover sketches in various domains.

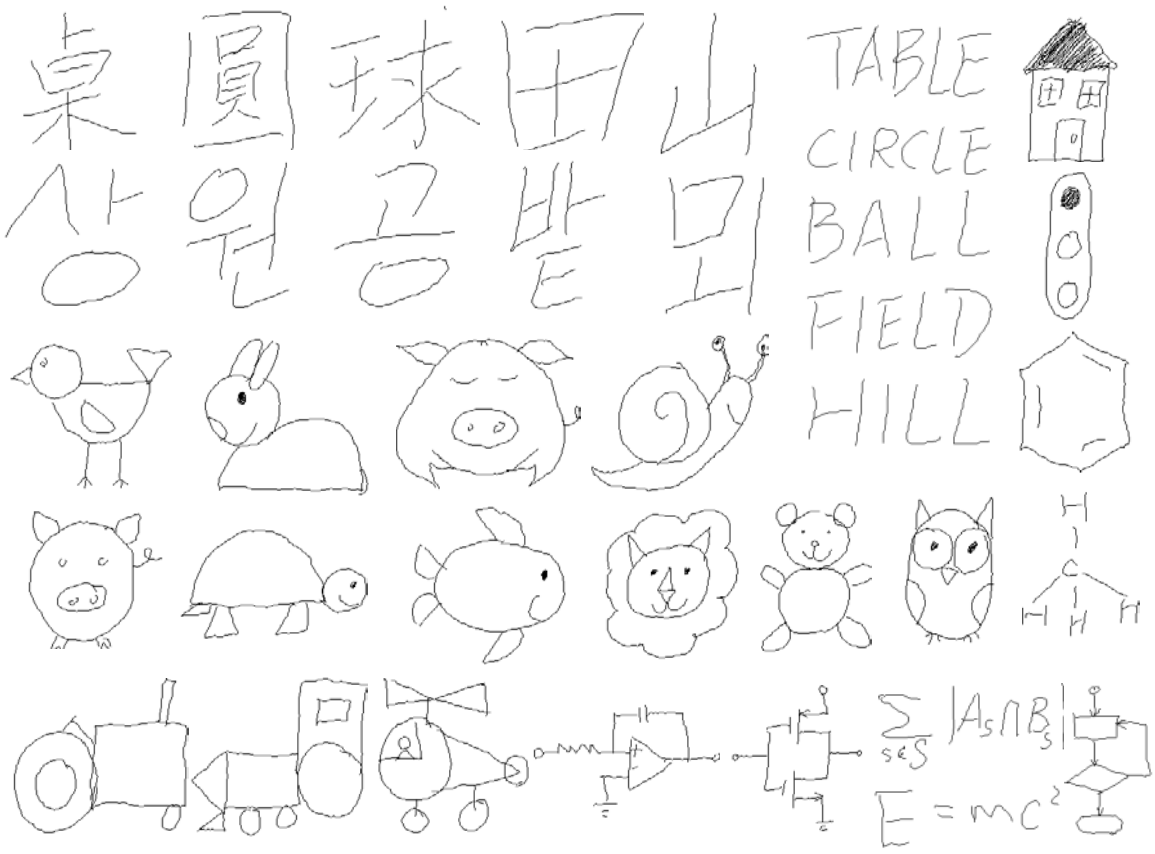


Figure 6 All 37 classes of sketches

Figure 7 shows a few examples of different classes of sketches drawn by several people.

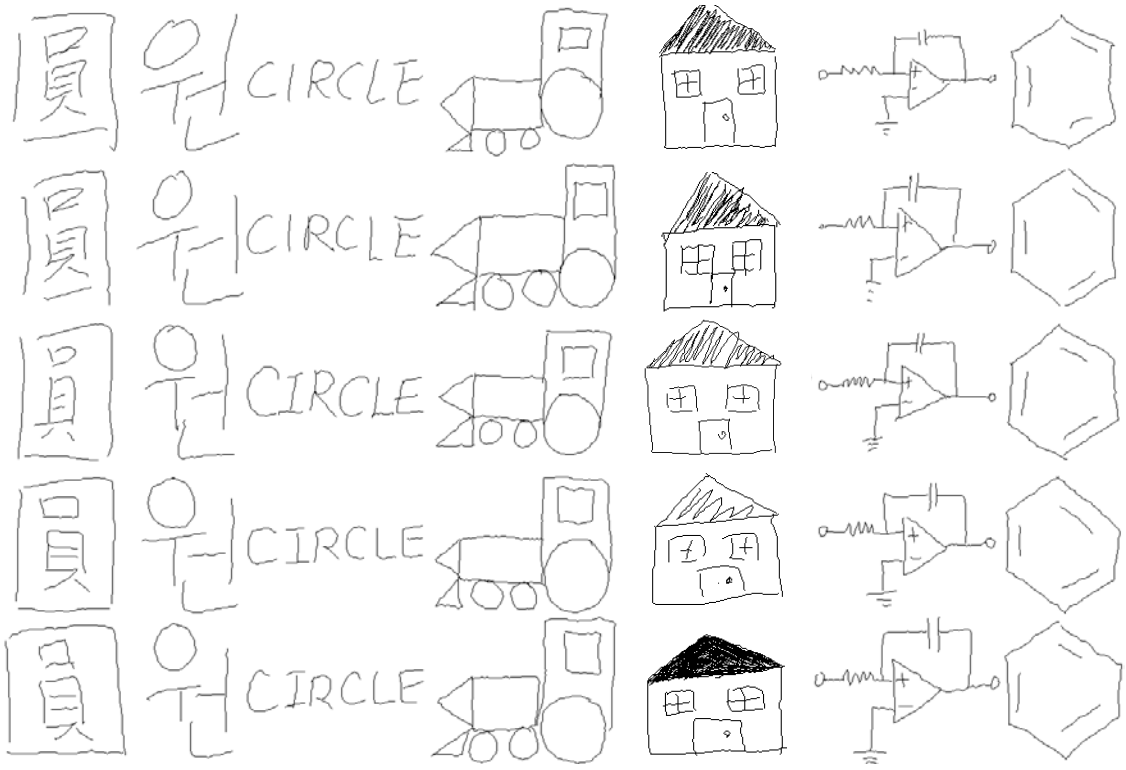


Figure 7 Some classes of sketches drawn by different people

2.5. Experiment Setup

In the database, there are 703 sketches in total. For each of the 37 classes of sketch, there are 19 sketches in the database that belong to that class. For each query, we retrieve the elements from the database in the descending order of similarity scores. For the experiment, each of the sketches in the database is used as the query to retrieve other sketches within the same class and the retrieval results are averaged within each class.

2.6. Evaluation Criteria

To evaluate the retrieval performance, the precision and recall graph [47] is plotted based on the ranks of those sketches from the same class as the query sketch. Figure 8 illustrates the concepts of recall and precision. The big circle indicates the retrieved sketches that are similar to the query sketch. *Recall* is defined as the ratio between the

number of relevant retrieved sketch and the total number of relevant sketches. *Precision* is defined as the ratio between the number of relevant retrieved sketch and the total number of retrieved sketch. When more items are retrieved, recall will be increased but precision will be decreased. A graph can be plotted with these recall-precision pairs and Figure 9 shows an example of such recall-precision graph. In each of our experiments, many queries are used and the resulting graph is obtained by averaging over all the queries. In a recall and precision graph, the higher the curve, the better the retrieval performance since for the same recall value, a higher curve signifies a higher precision value.

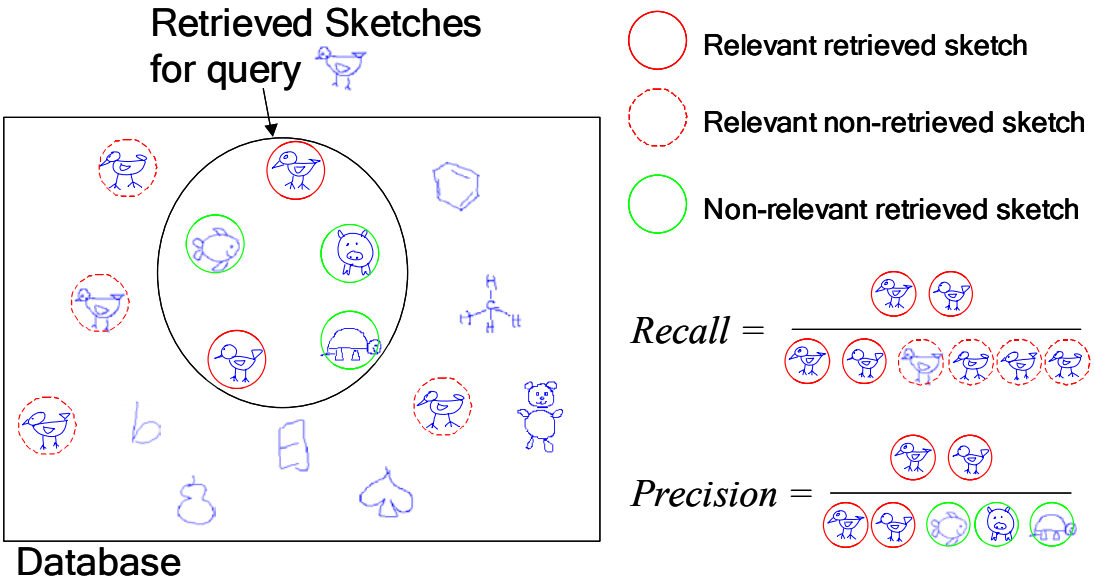


Figure 8 Illustration of recall-precision evaluation criteria

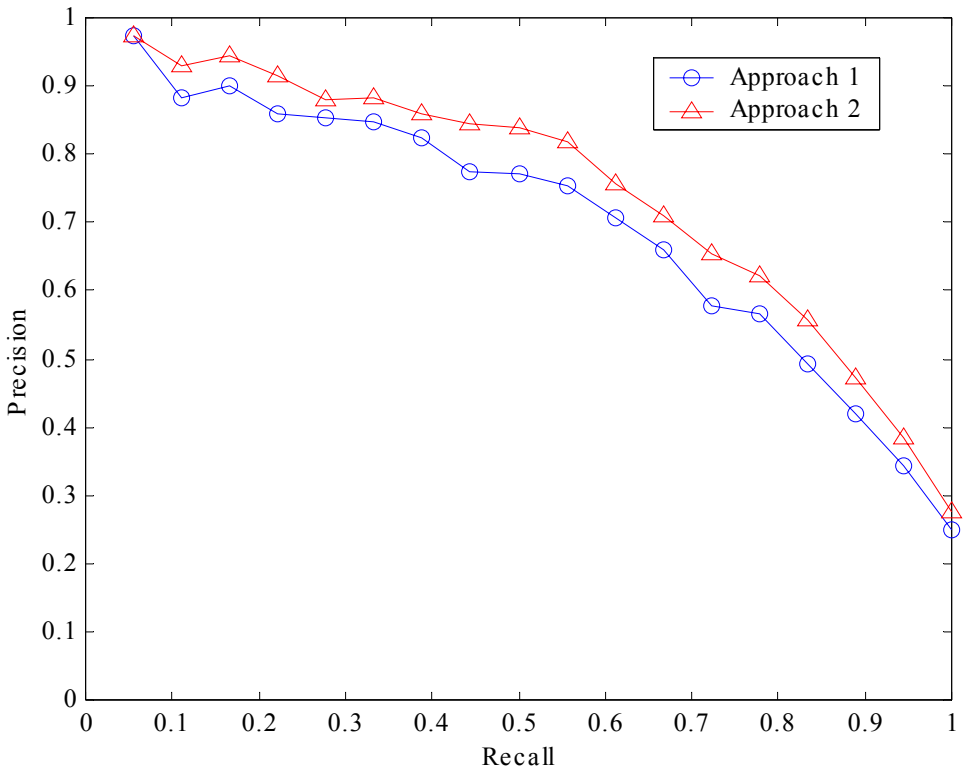


Figure 9 Recall-precision graph

3. Multiform Representations

This chapter focuses on the multiform representations part of the sketch retrieval system. The sketch is first resampled such that neighboring points in the same stroke have equal distance. Then we obtain the split representation by detecting the dominant points of the strokes and then obtain the merged representation by combining split strokes that are likely to form one of the primitive shapes.

Different people may draw the same sketch in a different way due to variations in style. Moreover, sometimes even the same person may draw the same sketch differently due to user inconsistency. In some shape recognition approach [27][48], dominant points are detected from the contour in order to be used as feature points. On the other hand, in handwriting recognition [26][45], similar method has been proposed to split or merge the strokes in order to have a better representation for improving the recognition result. However, variations in general sketches are much more than a specific domain of handwritings therefore it is unlikely to find a good criterion for splitting and merging the strokes to form a single consistent representation. As a result, we propose to use multiform representations for each sketch. From the original representation of a sketch, we try to split the strokes into smaller stroke segments based on the dominant points to obtain the split representation. Then from the split representation, we would like to merge the stroke segments if they form a primitive shape to become the merged representation. The connectivity of the strokes is first analyzed to divide the sketch into

components. Then for each component we search for the stroke segments that are likely to form a primitive shape.

The chapter is organized as follows. Section 3.1 explains the resampling process. Section 3.2 describes how to find the split representation based on the dominant point detection. Section 3.3 discusses how to find the merged representation by dividing the sketch into connected components and then going through the connectivity of the strokes.

3.1. Resampling

The sketch, consisting of a single stroke or multiple strokes, is resampled such that neighboring points in the same stroke have equal distance. Different people may draw at a different pace and this resampling process reduces inconsistencies due to different writing speed. For example, Figure 10 shows two sets of stroke samples for the same stroke. In the first example on the left hand side, the user starts drawing slowly and then draws faster. That's why the captured stroke samples are sparser at the beginning and become denser later. On the other hand, in the second example on the right hand side, the user starts drawing fast and then draws slowly. That's why the captured stroke samples are denser at the beginning and become sparser later. In order to account for the different drawing speed, the strokes are resampled so that after the resampling, successive stroke samples have the same distance as shown in Figure 10.

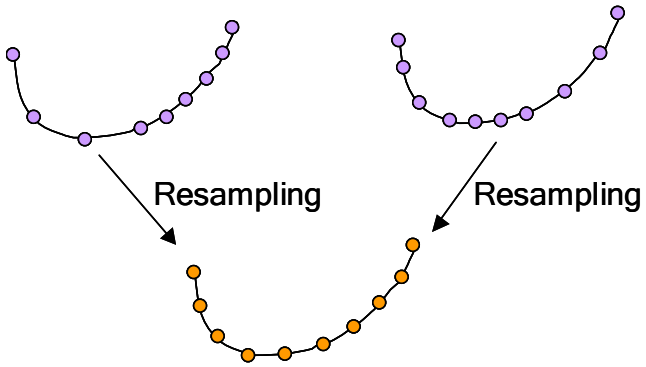


Figure 10 Equi-distance resampling

3.2. Splitting

The stroke samples are analyzed in order to detect for dominant points [2] that are points of interest indicating the locations of the stroke to be split. The signal that is used for dominant point detection can be illustrated by the system diagram in Figure 11. The resampled stroke samples ($x[n]$,$y[n]$) are passed in parallel into two systems with impulse responses $h_1[n]$ and $h_2[n]$. The norm of their difference is calculated to get the output signal that is the distance function $d[n]$.

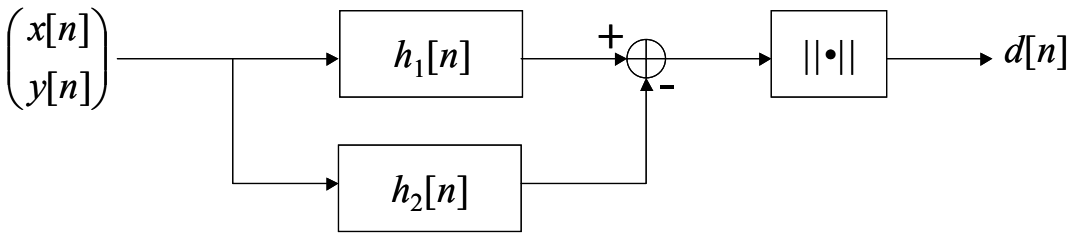


Figure 11 System diagram for getting the signal used for dominant point detection

Alternatively, the distance function $d[n]$ can also be expressed mathematically by equation (1):

$$d[n] = \sqrt{\left(x[n] \otimes h_1[n] - x[n] \otimes h_2[n]\right)^2 + \left(y[n] \otimes h_1[n] - y[n] \otimes h_2[n]\right)^2} \qquad (1)$$

The impulses responses $h_1[n]$ and $h_2[n]$ are Gaussian windows with different window sizes W_1 ,W_2 and variances σ_1 ,σ_2 as specified in equation (2). As a result,

convolving with $h_1[n]$ and $h_2[n]$ means that $x[n]$ and $y[n]$ are smoothed with two different scales.

$$h_i[n] = \begin{cases} \dfrac{1}{K_i\sqrt{2\pi\sigma_i}} e^{-\frac{n^2}{2\sigma_i^2}} & |n| \le W_i \\ 0 & |n| > W_i \end{cases} \qquad i = 1,2 \tag{2}$$

$$K_i = \sum_{m=-W_i}^{W_i} \frac{1}{\sqrt{2\pi\sigma_i}} e^{-\frac{m^2}{2\sigma_i^2}} \qquad i = 1,2 \tag{3}$$

The distance function thus measures how much each stroke sample is changed between the two smoothing operations. Based on this distance function, we try to detect the dominant points. The algorithm for detecting for the dominant points is illustrated by the flow chart in Figure 12. The sample points that are close to the two end points will never be considered as dominant points therefore we start with the sample far enough from one of the end points. The distance function evaluated at this sample is compared against a threshold. If it is greater than the threshold and if no dominant points were detected before, then this sample is considered as a new dominant point. On the other hand, if the distance is greater than the threshold but dominant points do exist, then the current sample index will be compared with the sample index of the last detected dominant point. If the two sample indices are very close, then only one of them will be kept as the dominant point. In particular, the sample index with a larger distance will be stored. Alternatively, if the two sample indices are not close, then the current sample index will be considered as a new dominant point. Afterwards we consider the next sample and continue this algorithm until the sample is close to the other end point.

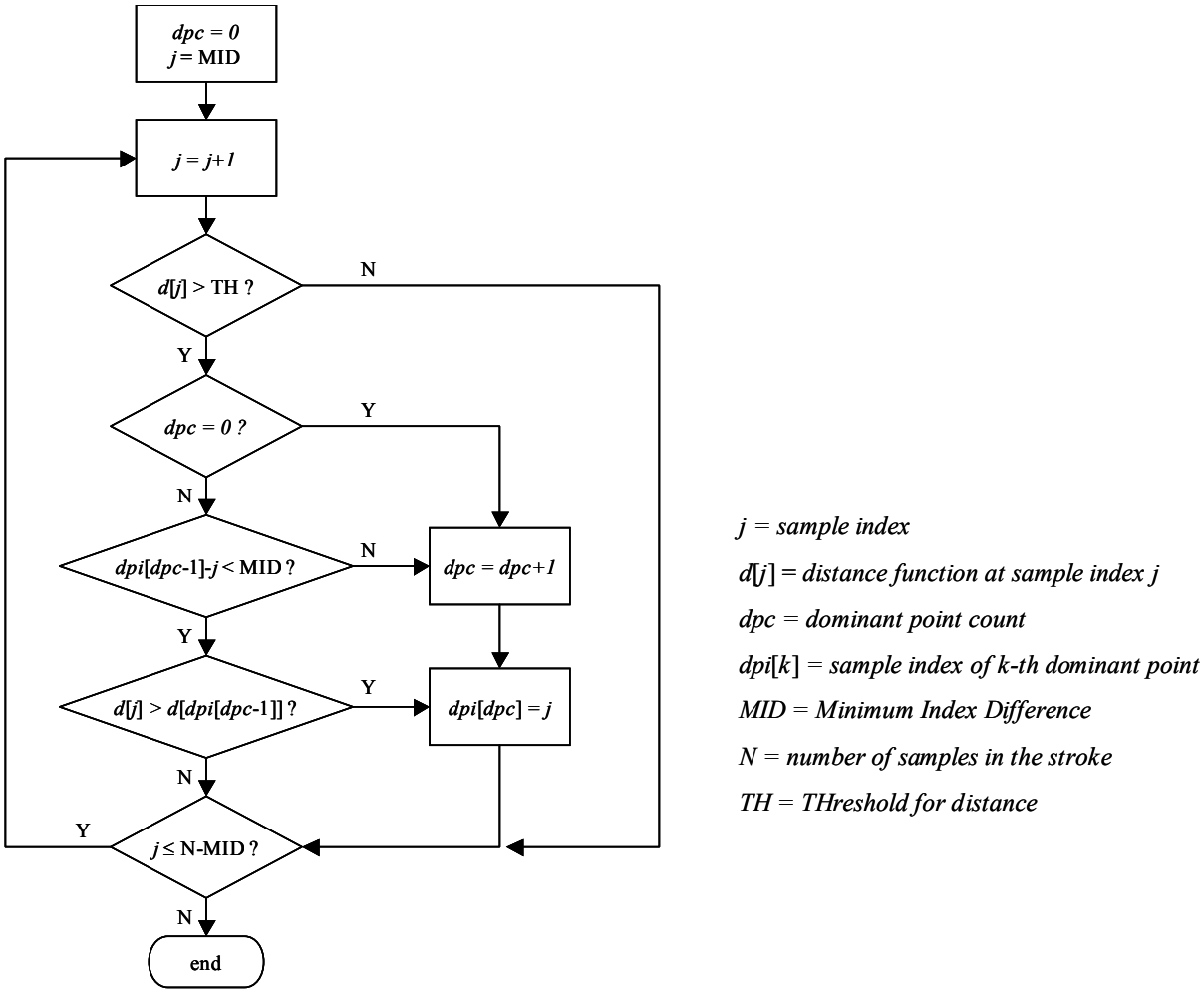


Figure 12 Flow chart for dominant point detection

Figure 13 shows some example sketches after splitting. It can be seen that a stroke indicated by the connected samples can be split into several stroke segments indicated by different colors. From the first example of Figure 13, it can be seen that the stroke is not only split on sharp corners, but is also split on points with smoother direction change. Although the first and second examples in Figure 13 correspond to the same sketch, they are drawn by different people and the variation in style is clearly visible. The sketches after splitting allow a more consistent representation that facilitates

matching at the later stage. The third and fourth examples in Figure 13 also provide other example sketches after splitting.

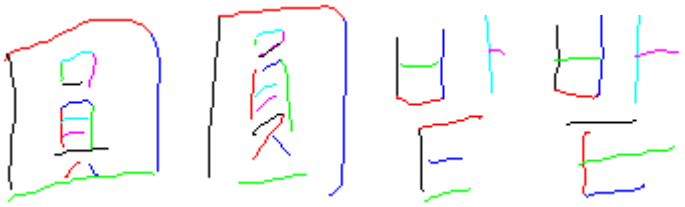


Figure 13 Example sketches after splitting

3.3. Merging

After splitting, we would like to merge the stroke segments if they form a primitive shape to become the merged representation. We define a primitive shape to be a circle, a polygon or a line. The connectivity of the strokes is first analyzed to divide the sketch into components. Then for each component we search for the stroke segments that are likely to form a primitive shape.

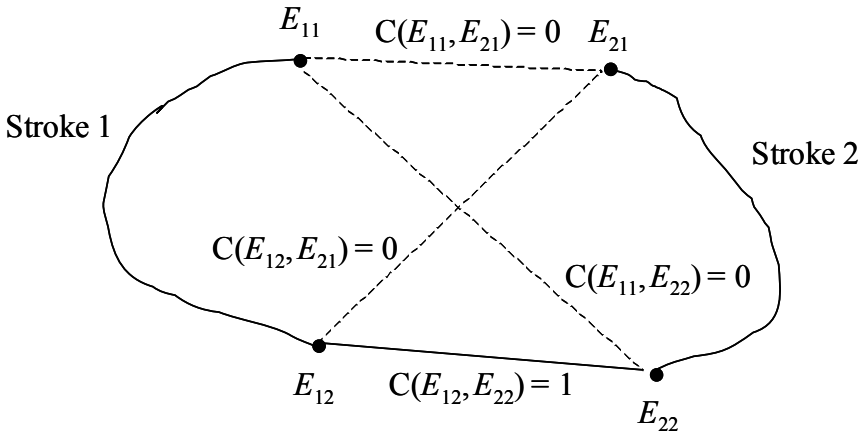


Figure 14 Connectivity between a pair of strokes

Each stroke segment has two end points. For a pair of strokes as shown in Figure 14, let E_{11}, E_{12} denote the end points of stroke 1 and E_{21}, E_{22} denote the end points of stroke 2. Further, let $C(E_a,E_b)$ denotes the connectivity between the end points E_a and E_b,

where $C(E_a,E_b) = 1$ means that E_a and E_b are connected and $C(E_a,E_b) = 0$ means that E_a and E_b are not connected. Stroke 1 and stroke 2 are said to be *merged* if one of the quantities from the set $\{C(E_{11},E_{21}),C(E_{11},E_{22}),C(E_{12},E_{21}),C(E_{12},E_{22})\}$ is equal to one. Since the connectivity takes a binary value, and there are 4 quantities associated the connectivity between the end points with a pair of strokes, therefore there are 2^4 ways of connecting 2 strokes. Assume that there are N stroke segments in the split representation. Then there are $\binom{N}{2}$ pairs of stroke segments. As a result, there is a total of $\left(2^4\right)^{\binom{N}{2}} = 2^{2N(N-1)}$ ways of connecting N strokes. Since this number increases exponentially as N increases, the search space for the merged representation will be huge and it is not feasible to perform exhaustive search. As a result, we need to find an efficient algorithm to search for a good merged representation.

We can first divide a sketch into components based on the end point proximity. Two strokes are considered as the same component if there exists a pair of end points between these strokes that are close. There is a tradeoff in choosing the threshold for the end point proximity. When the threshold is too small, then the number of strokes in the same component will be smaller, thus reducing the complexity of the search space. However, this also means that it is more sensitive to user variation since some people may leave a larger space between neighboring strokes. On the other hand, when the threshold is too large, it is less sensitive to user variation but at the same time the number of strokes in the same component may be large such that it may defeat the purpose of dividing the sketch into components.

After grouping the strokes into components, we need to find out whether some stroke segments can be merged to form a primitive shape. For examples, line segments may be merged if they form a polygon or arc segments may be merged if they form a circle. The polygon and circle primitive shapes are both characterized by a closed contour. As a result, we perform the following 2 steps in order to decide which stroke segments to be merged:

1) find stroke segments that form closed contours within a component

2) merge those stroke segments if the resulting closed contour is likely to be a primitive shape

The likelihood of primitive shapes will be discussed in the Section 4.5. Here we focus on step 1 and explains our algorithm for closed contour detection from a set of line segments. First we determine the existence of closed contour within a set of stroke segments. A closed contour composed of several stroke segments has the property that each end point E_{i1}, E_{i2} of each stroke segment is connected to an end point from another stroke segment of this closed contour. As a result, given a set of stroke segments, in order to detect for a closed contour, we first find a subset of stroke segments that have the property that both end points of each stroke segment in this subset are connected to some other end points. Then we discard those stroke segments whose neighboring stroke segments are not in the subset. For example, as shown in Figure 15, there are six stroke segments. Based on our closed contour detection algorithm described above, we pick up a subset of stroke segments {stroke 1, stroke 2, stroke 3, stroke 5}of which both their end points are connected to some other end points. We then discard stroke 5 from this subset because at least one of its neighboring stroke (stroke 4 or stroke 6) is not in this subset.

The modified subset becomes {stroke 1, stroke 2, stroke 3} which contains at least one closed contour. After this step, the number of stroke segments is reduced and we can trace the stroke segments in order to locate the closed contours.

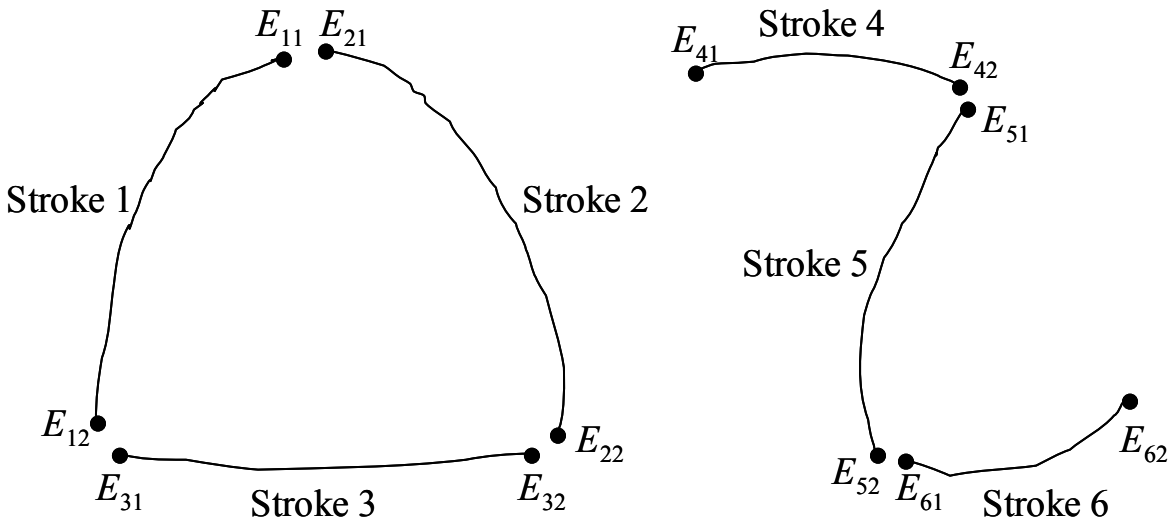
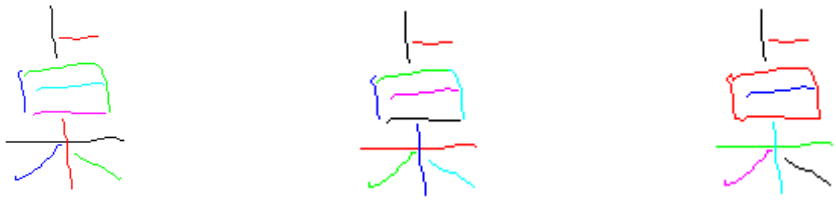


Figure 15 Example strokes for demonstrate closed contour detection

Figure 16 shows the comparison between multiform representations. It can be seen that a square is detected in the merged representation whereas it consists of 3 strokes in the original representation and is broken down into 4 stroke segments in the split representation.



(a) Original Representation (b) Split Representation (c) Merged Representation

Figure 16 Comparison between the multiform representations

3.4. Experiment and Result

We perform an experiment to analyze the gain in the retrieval performance if we use multiform representations instead of each representation alone. In this experiment, shade detection specified in Section 4.1 is first applied to the sketch after resampling in order to represent each shaded region by only one basic stroke such that the number of strokes is reduced for each multiform representation. The shape features of the strokes for each representation are then extracted according to Section 4.5. The correspondence of the stroke features between the query and each sketch in the database is determined by the Hungarian method specified in Section 5.1. With the correspondence, the similarity score for that representation is computed by combining the shape feature similarity and the spatial relation similarity as described in Section 5.2.5. Afterwards, the similarity score for each representation is normalized according to the mean similarity score for that representation and then the normalized similarity scores for all representations are linearly combined to give the similarity score for multiform representations as described in Section 5.2.5. Figure 17 shows the comparison of the retrieval performance for this experiment. It can be seen that the retrieval performance is higher for multiform representations than for each representation alone.

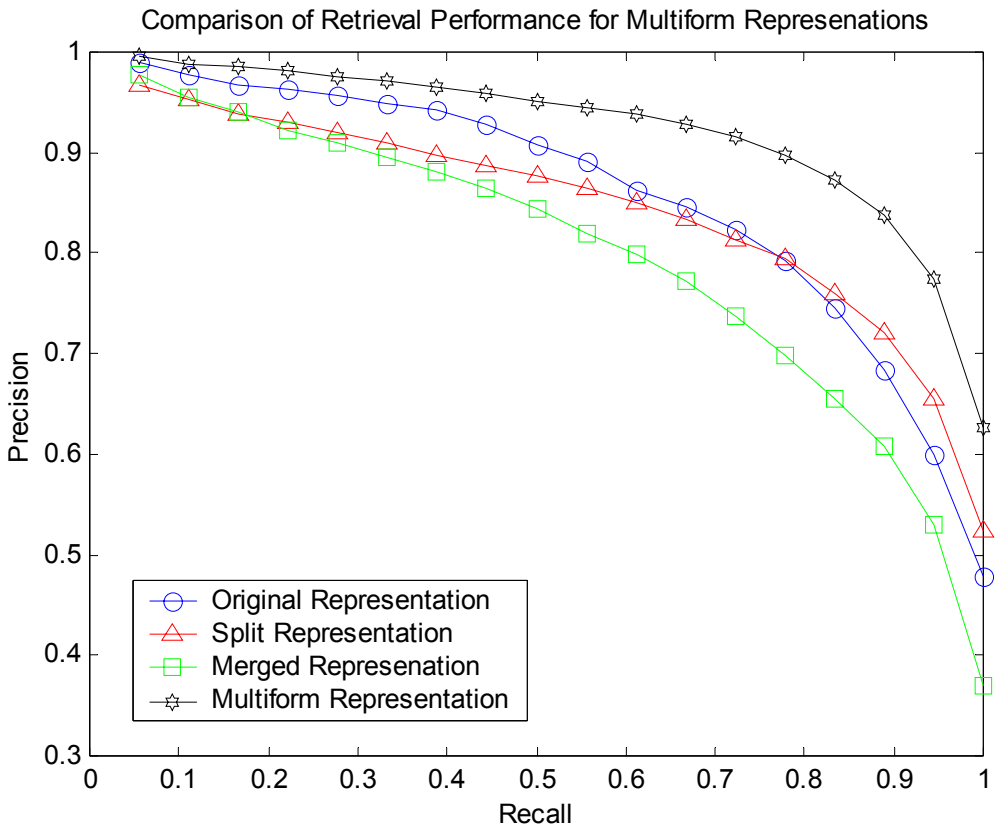


Figure 17 Retrieval performance with multiform representations

In Figure 17, it can also be observed that the merged representation performs the worst among the three representations in the overall sense as the result is obtained by averaging over all classes of sketches. Now we provide some case studies to show that each representation can perform well for a certain class of sketches.

Case 1: Example class that works well for the original representation

In Figure 18, the multiform representations for class 4 contain sketches of a Chinese character. The color indicates the stroke connectivity. It can be seen that the sketches in the original representation are more consistent (the sketches have similar break points) since most of the subjects who created the sketches in our database know the Chinese language and they follow the rules in writing the character. This explains

why the retrieval performance is better for the original representation for this class as indicated in Figure 19.

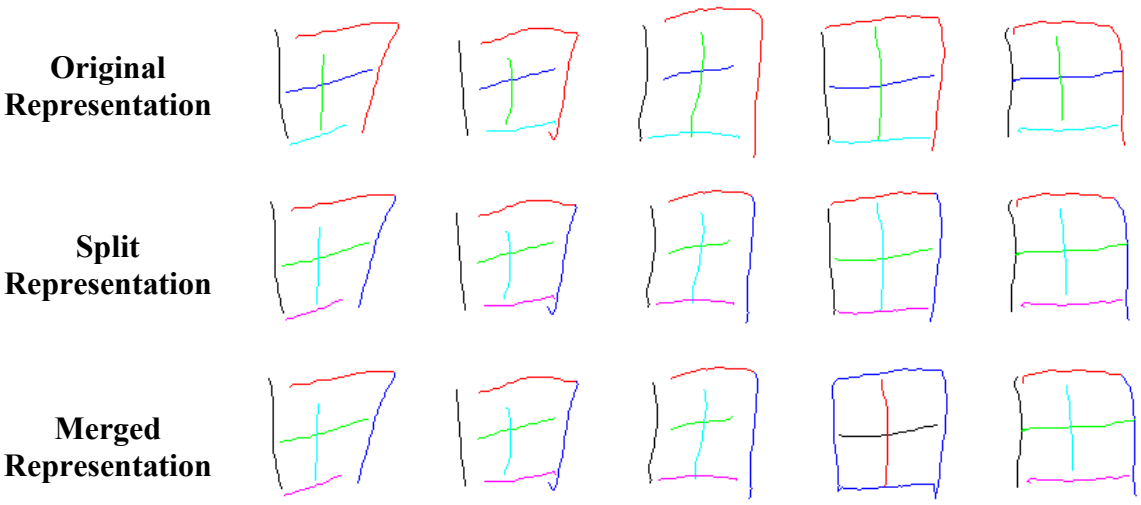


Figure 18 Multiform representations for some sketches in class 4

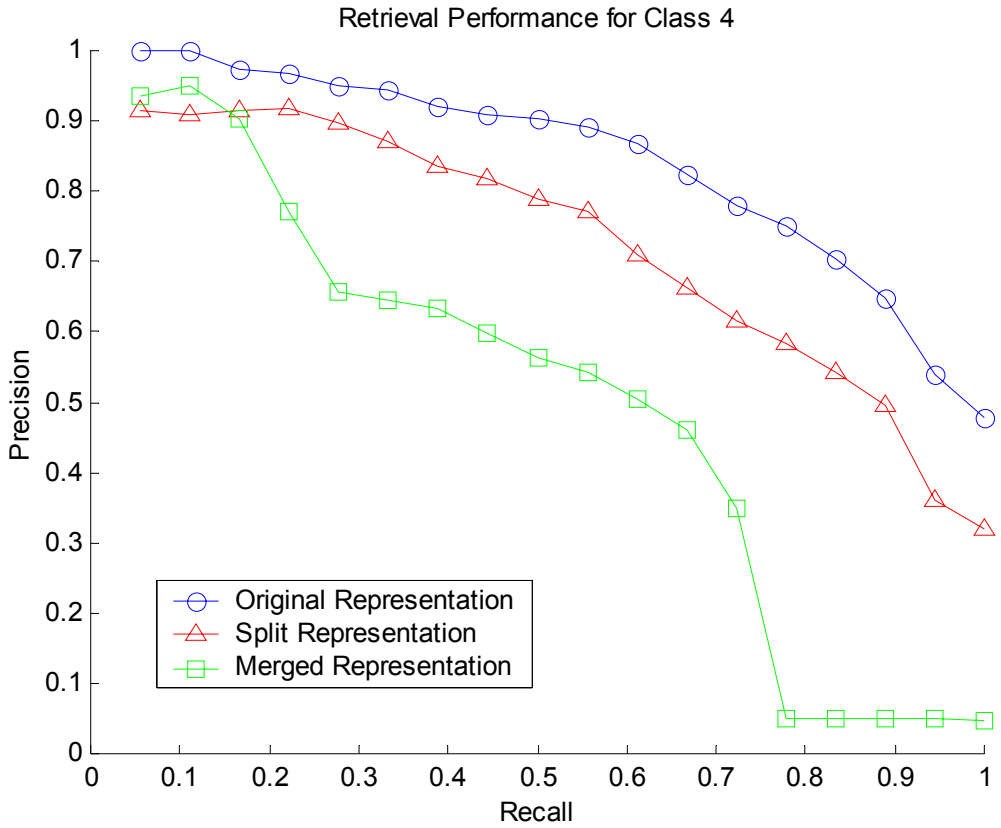


Figure 19 Retrieval performance for Class 4

Case 2: Example class that works well for the split representation

In Figure 20, the multiform representations for class 36 contain sketches of a chemical structure "benzene". It can be seen that the sketches in the original representation are not so consistent (the hexagon can be formed from 1 stroke, 2 strokes, or 6 strokes). On the other hand, the split representation becomes much more consistent for this class after breaking the hexagon into line segments. This explains why the retrieval performance is better for the split representation for this class as indicated in Figure 21. The merged representation for this class also performs better than the original representation but is slightly worse than the split representation. This is because sometimes the hexagon is not detected when the distance between the stroke end points is too big as indicated in the 4th column of Figure 20.

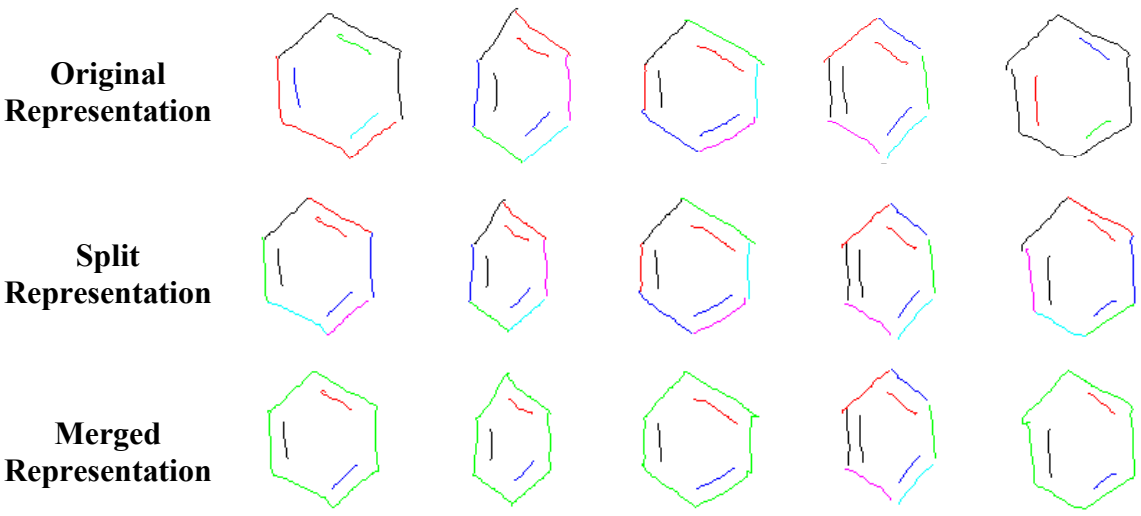


Figure 20 Multiform representations for some sketches in class 36

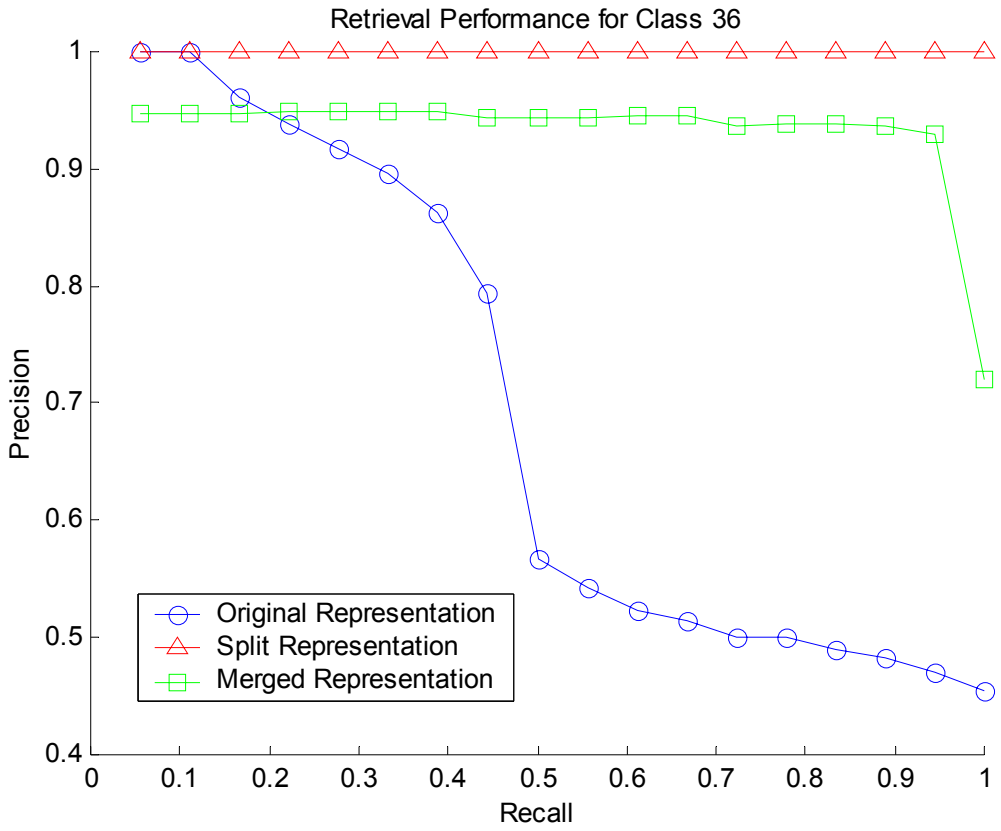


Figure 21 Retrieval performance for Class 36

Case 3: Example class that works well for the merged representation

In Figure 22, the multiform representations for class 6 contain sketches of a Korean character. It can be seen that the sketches in the original representation are not so consistent (the inverted V-shaped on the top left part sometimes consists of one stroke but sometimes it is written as two strokes and sometimes it may even look different). Since most of the subjects who created the sketches in the database do not know the Korean language, the rules about writing Korean characters may not be followed so it explains the inconsistency of the original representation as opposite to Case 1. On the other hand, the split representation makes this inverted V-shaped more consistent by always breaking it into 2 strokes. However, the circle at the bottom is also split into segments at various

locations since it is an imperfect circle. After merging, the stroke segments are merged back to form the circle again. This explains why the retrieval performance is better for the merged representation but worse for the split representation for this class as indicated in Figure 23.

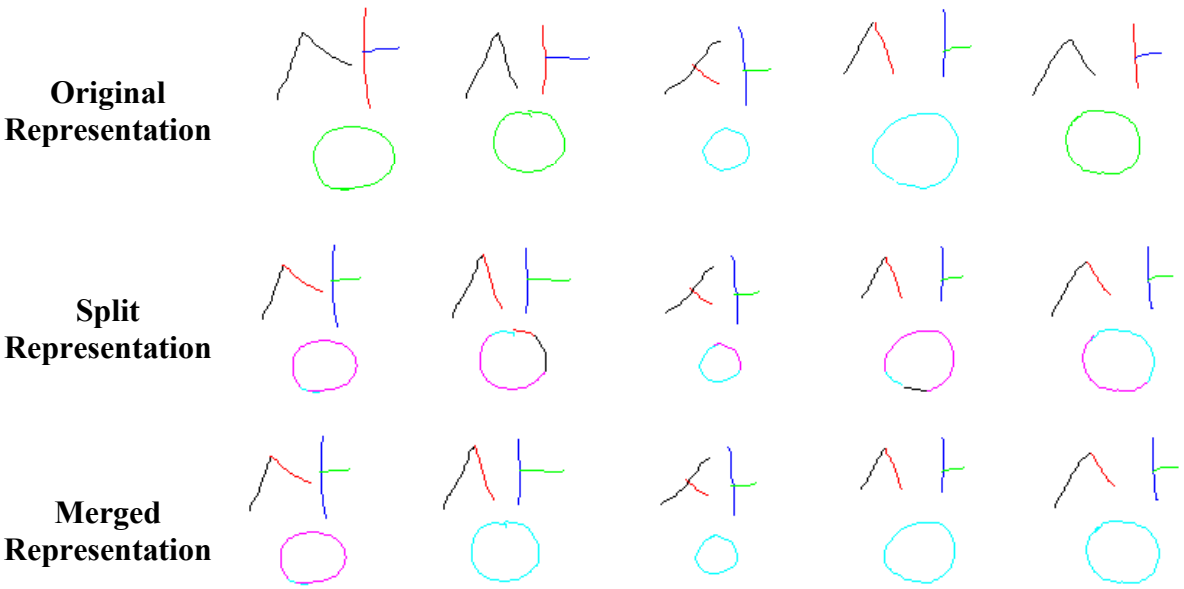


Figure 22 Multiform representations for some sketches in class 6

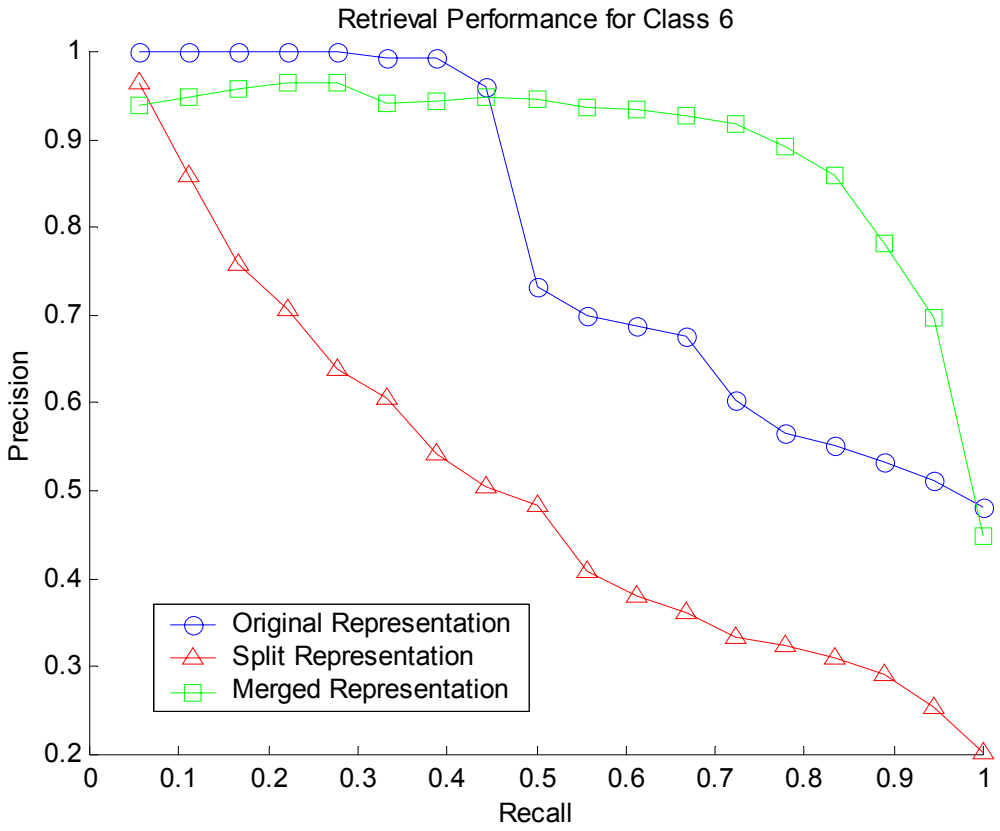


Figure 23 Retrieval performance for Class 6

4. Coarse-To-Fine Feature Extraction

This chapter focuses on the feature extraction module in a coarse-to-fine manner. Coarse-to-fine feature extraction can be conceptually considered as looking at the sketch at different points of view. When coarse features are extracted, it can be considered as the case when the sketch is being viewed far away and when fine features are extracted, it can be considered as the case when the sketch is being viewed nearby.

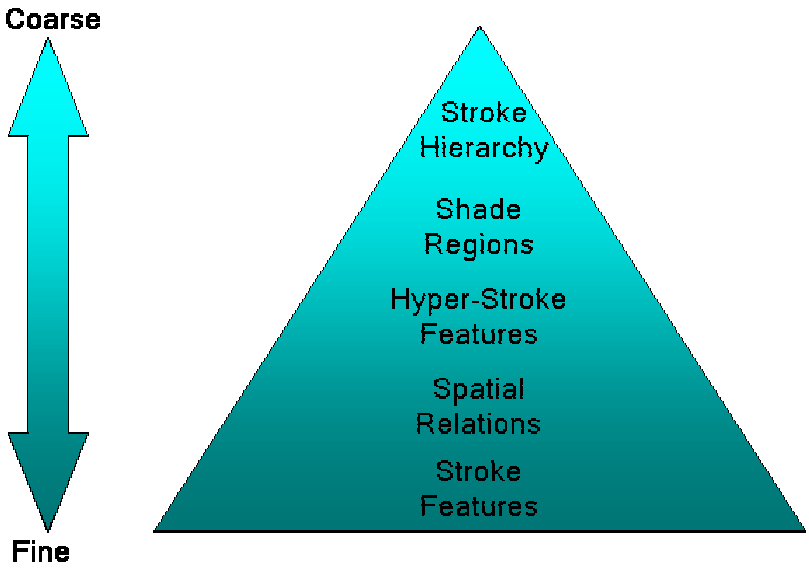


Figure 24 Different levels of features

The choice of features is often based on heuristics and it depends on the data. We propose a to extract features in a coarse-to-fine manner such that features at different levels are considered. As a result, in addition to low-level features, we also extract high-level semantic features from the representation after the preprocessing stage. We propose several strategies in the coarse-to-fine feature extraction that is illustrated in Figure 24. First the strokes are analyzed in order to detect for shaded regions since the strokes in a

shaded region are better treated as a single unit. Then we construct the stroke hierarchy of each sketch to capture the structural information. We introduce a novel concept of *hyper-stroke* that consists of a set of strokes smartly grouped together. We then approach the sketch in a finer detail when the spatial relations between the strokes are considered. Finally for basic strokes, we describe them with features of several primitive shapes: line, polygon and circle. The likelihood of each stroke that falls into a primitive shape is estimated.

The chapter is organized as follows. Section 4.1 describes the shade detection for sketches. Section 4.2 explains how to construct the stroke hierarchy of a sketch. Section 4.3 introduces a novel concept of hyper-strokes that characterize groups of strokes. Section 4.4 specifies what kind of spatial relations between the strokes to be considered in our approach. Section 4.5 provides an explanation about the primitive shape features of basic strokes. Section 4.6 shows a simplified algorithm for shade detection under certain assumption and it also shows how the idea of shade detection can be applied for images.

4.1. Shade Detection

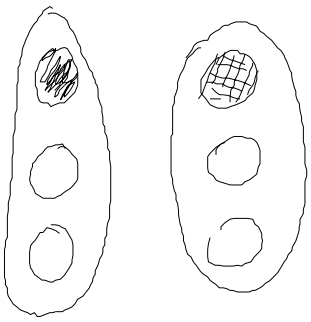


Figure 25 Example sketches with shaded regions

For a shaded region, a user may draw many strokes to describe the shade. It is better to consider this shaded region as one unit instead of considering each stroke separately. This is because two sets of strokes forming the same shaded region may not look similar as explained before with the examples in Figure 25. As a result, a shaded region is represented by a single unit containing features describing the shaded region. A region is more likely to be a shaded region if the ink density of the strokes in that area is high. Starting with this observation, we develop the algorithm for the shade detection according to the following steps.

4.1.1. Shade Detection Algorithm

Step 1: Detect blocks with high ink density

The canvas containing the sketch is divided into blocks of $M \times M$ pixels. For each block, the number of stroke samples is counted. If this number is not smaller than a certain threshold TH_{num_sample}, then that block is considered to be with high ink density.

Step 2: Detect regions with large number of neighboring blocks with high ink density

Neighboring blocks are combined in order to form a connected region. If the number of blocks in a connected region is not larger than a certain threshold TH_{num_block}, then that connected region is ignored. Otherwise this region is considered to be with high ink density.

Step 3: Find those strokes lying inside the convex hull of each connected region

For each connected region with high ink density, the convex hull of this connected region $CONV\_HULL_{region}$ is constructed from the four corners of each of all the blocks forming the connected region. Each stroke is being compared with the convex

hull of the connected region. If the percentage of the stroke samples inside the convex hull is greater than a certain threshold $TH_{percent\_sample\_in\_conv\_hull}$, then that stroke is considered to be inside the convex hull of that connected region.

Step 4: Classify the strokes into boundary or non-boundary strokes

For each stroke that is inside the convex hull of a connected region, the distance from each sample point of that stroke to the closest line segment of the convex hull boundary is computed and the average distance over all stroke samples is calculated. If this average distance is smaller than a certain threshold $TH_{average\_distance}$, then that stroke inside the convex hull is classified as a boundary stroke. Otherwise, it is classified as a non-boundary stroke.

Step 5: Construct the convex hull of all boundary strokes and non-boundary strokes for each region

For each region, the convex hull $CONV\_HULL_{stroke}$ containing all boundary strokes and non-boundary strokes is constructed. Essentially this convex hull $CONV\_HULL_{stroke}$ refines the shaded region compared with the convex hull of the connected region $CONV\_HULL_{stroke}$ obtained in Step 3.

Step 6: Replace the points on the boundary of $CONV\_HULL_{stroke}$ by the nearest point from a boundary stroke if their distance is small enough

For each corner of $CONV\_HULL_{stroke}$, the distance between this corner and the nearest stroke sample of a boundary stroke is computed. If this distance is smaller than a certain threshold $TH_{corner\_distance}$, then the corner is replaced by that nearest stroke sample of the boundary stroke. This step takes care of the case when the boundary stroke exists

for a shaded region but the shade is drawn such that some of the strokes fall outside the boundary. An example of this case is shown in Figure 26. The boundary stroke that has been drawn by the user is indicated in red and all other strokes for this shade are indicated in cyan.

Figure 26 Shaded region with part of the shade falls out of the boundary

Step 7: Compute the percentage of stroke samples from each boundary stroke that contribute to the modification of convex hull boundary.

If the percentage is less than a certain threshold $TH_{percent\_sample\_in\_boundary}$, then this boundary stroke is not considered as boundary stroke any more and it will be labeled as a non-boundary stroke. This step takes care about the case if there exist two or more boundary strokes that are close to the same portion of the shaded region boundary, only the nearest boundary stroke is considered to contribute to the true boundary.

Step 8: The final boundary stroke for that shaded region is replaced by the modified convex hull boundary obtained from Step 6 and all remaining boundary strokes are discarded.

Step 9: The resulting boundary stroke and non-boundary strokes can be considered as a hyper-stroke and its hyper-stroke features are extracted as described in Section 4.3. In

addition, the boundary stroke is treated as a basic stroke to extract the shape features as described in Section 4.5.

The advantage of our algorithm is that it does not have the assumption that the user must include a boundary when he/she draws a shaded region. Nevertheless, if this assumption is valid, then we have a simplified algorithm for the shade detection as described in the Appendix A in Section 4.6.1. In addition, the idea of shade detection can also be applied to images and the extension is provided in the Appendix B in Section 4.6.2.

4.1.2. Shade Detection Parameters

Since the size of the shade region varies, therefore we apply the algorithm specified in the previous section using 3 sets of parameters in order to detect the shade at different scales. The parameters are shown in Table 1. Each column represents a set of parameters at one scale. The ink density of the block at the threshold for deciding high vs. low ink density can be calculated by with equation (4):

$$TH_{ink\_density} = \frac{TH_{num\_sample}}{M \times M} \tag{4}$$

For the first set of parameters , TH_{num_sample} = 0.63. For the second set of parameters, TH_{num_sample} = 0.31 and for the third set of parameters, TH_{num_sample} = 0.11. On the other hand, the threshold of the number of neighboring high ink density blocks TH_{num_block}, for deciding whether a connected region has high ink density, is the smallest for the first set and is the largest in the third set. As a result, it can be seen that the first set of parameters is used to detect for shade at a finer scale by looking for small regions

with very high ink density. On the other hand, the third set of parameters is used to detect for shade at a coarser scale by looking at larger regions with a more relaxed constraint of high ink density measure. The second set of parameters is used to detect for shade at a medium scale in-between the first set and the third set. Some examples of shade regions with different sizes are shown in Figure 27.

Parameter	Value for the 1st Scale	Value for the 2nd Scale	Value for the 3rd Scale
M	4	4	8
TH_{num_sample}	10	5	7
TH_{num_block}	0	3	13
$TH_{percent\_sample\_in\_conv\_hull}$	65%	65%	65%
$TH_{average\_distance}$	5.0	5.0	7.0
$TH_{corner\_distance}$	9.0	9.0	9.0
$TH_{percent\_sample\_in\_boundary}$	70%	70%	70%

Table 1 Shade Detection Parameters

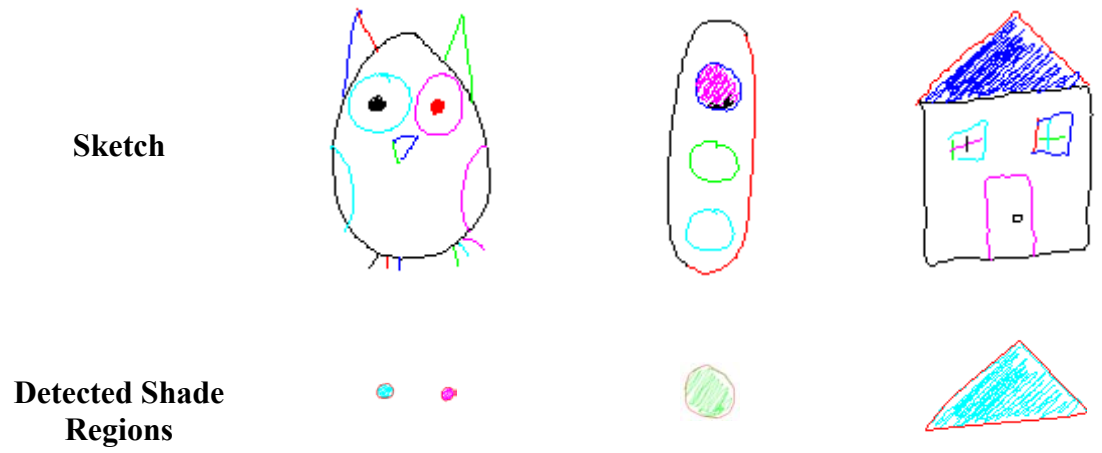


Figure 27 Shaded regions with different sizes

4.2. Stroke Hierarchy Construction

Representing a sketch by a stroke hierarchy can be treated as a multiple scale representation. In [20], multiple scale representation is obtained by applying successive morphological operations (opening or closing) to tumor shapes and the morphological

distance is computed by combining distances at different scales. The wavelet representation in [16] can also be considered as a hierarchy in the spatial frequency domain. Here the difference in wavelet coefficients is used as a distance measure. The hierarchical representation in our system is based on the spatial hierarchy of the strokes. Each stroke is considered as one component and the matching is performed in a stroke-by-stroke basis in order to better tolerate local variations. Furthermore, a clustering method such as ACE (Aggregation Clustering with Exceptions) [52] may be used to extract an effective storage structure given the instance relationships from the stroke hierarchy.

Now we introduce our approach for building a stroke hierarchical representation for a sketch. The relationship between a parent stroke and a child stroke is that the child stroke is *inside* the parent stroke. The stroke hierarchy describes the structural relationship between the strokes in a sketch. For example, in drawing a house, the windows and the door are drawn inside the front part of the house; and the doorknob is drawn inside the door. As a result, the corresponding stroke hierarchy is the one shown in Figure 28. In the actual implementation of this module, a quick bounding box test will be first performed between every pair of strokes to determine whether their bounding boxes overlap. If there is a significant amount of overlap between their bounding boxes, then a further convex hull test will proceed. In the convex hull test between two strokes, if most of the samples of a stroke fall inside the convex hull bounding the other stroke, then the first stroke is considered to be inside the second stroke. Under this rule, when there are two overlapping strokes, it is possible for each stroke to be considered inside the other stroke simultaneously. In this case, we will cancel out their effect by removing the

"inside" relationships on both sides. After getting the "inside" relationship between the strokes, these relationships are examined to get the hierarchical information. For example, if stroke 3 and stroke 4 are both inside stroke 2; and if stroke 4 is also inside stroke 3, then stroke 2 is considered as the parent stroke of stroke 3; and stroke 3 will be considered as the parent stroke of stroke 4. A sketch of house and its corresponding stroke hierarchy are shown in Figure 28.

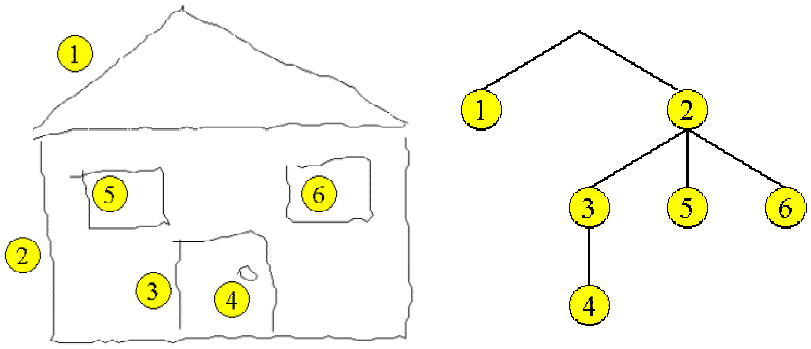


Figure 28 A sketch and its corresponding stroke hierarchy

4.3. Hyper-Stroke Feature Extraction

Under traditional image processing, a pyramid representation of an image with multiple levels of detail can be obtained by filtering the image several times. For sketches, instead of using the filtering techniques that rely on preprocessing low-level features, we propose to obtain multiple levels of detail by considering groups of strokes (thus in a higher semantic level) in addition to the basic strokes alone. We define *hyper-stroke* as a group of strokes such that there exists one basic stroke describing the boundary and all the strokes (at least one) inside the region enclosed by the basic stroke. Based on this definition, the shaded region described in Section 4.1 can also be represented by a hyper-stroke where the basic stroke of the hyper-stroke is the boundary stroke of the shaded region and there exists at least one non-boundary stroke enclosed by

the boundary stroke in the shaded region. In addition, a hyper-stroke can also be derived from the stroke hierarchy. Given the stroke hierarchy of a sketch, a hyper-stroke is formed by a parent stroke with all its descendent strokes with the parent stroke as the basic stroke. A hyper-stroke should contain at least one descendent stroke therefore the stroke at a leaf node that does not enclose any other strokes is not a hyper-stroke. The relationship between the stroke hierarchy and a hyper-stroke is illustrated in Figure 29. The hyper-stroke features are extracted for all the strokes inside the region enclosed by the basic stroke. We extract the hyper-stroke features as the Hu moments [15] and the histogram of edge directions that are commonly used region-based features. As a result, hyper-stroke features provide an overall description about a region in a sketch. This is analogous to the region-based approaches used for image retrieval [6][17].

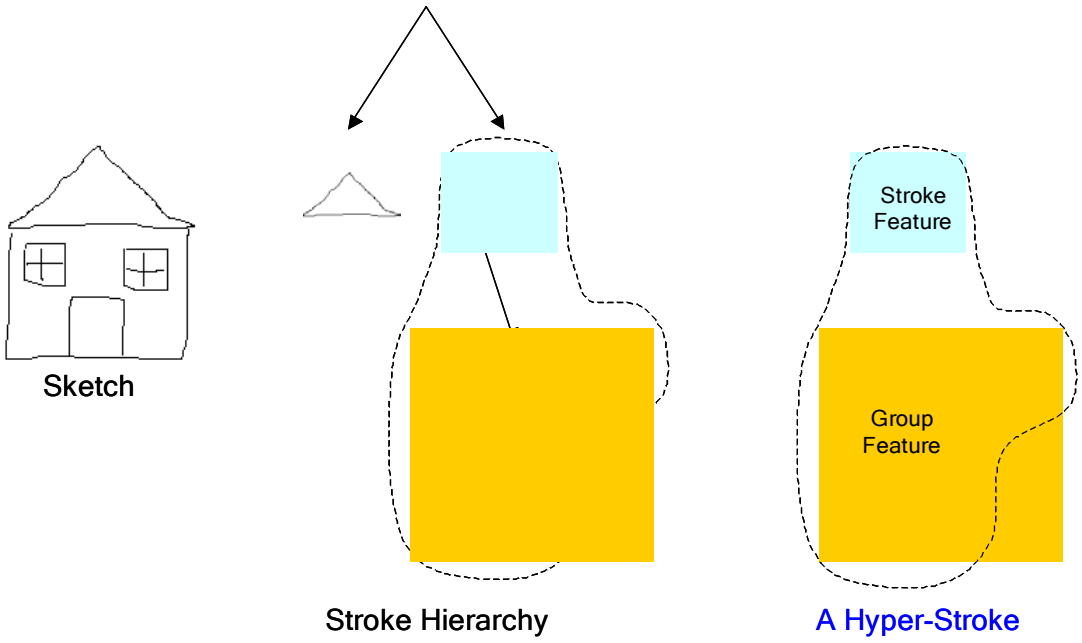
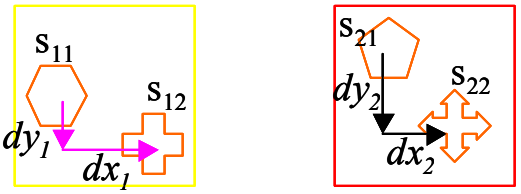


Figure 29 Relationship between a stroke hierarchy and a hyper-stroke

4.4. Spatial Relations

Several approaches exist for describing spatial relations between two components of an object. One approach is to compare the spatial ordering using some reference locations such as the centroid or a corner of the bounding rectangle of the components [51]. Another approach is to consider the relationship between the intervals defined by the bounding rectangle of the components [35]. The above approaches both quantize the spatial relations into a few discrete states, therefore there may be a big jump in spatial relations even when the actual change in spatial location of a component is small. As a result, we propose to model the spatial relation with a continuous function to avoid non-proportional change between spatial relation and spatial locations.

We compute the displacement vector between the two distances of the corresponding stroke pairs as a measure of spatial relations. For example, as shown in Figure 30, the corresponding strokes for s_{11} and s_{12} of sketch 1 are s_{21} and s_{22} of sketch 2 respectively. The spatial relation between strokes s_{11} and s_{12} can be denoted by R_1 that is defined by the horizontal and vertical displacements (dx_1, dy_1) between the center of s_{11} and the center of s_{12}. Similarly, the spatial relation between strokes s_{21} and s_{22} can be denoted by R_2 that is defined by the horizontal and vertical displacements (dx_2, dy_2) between the center of s_{21} and the center of s_{22}. The spatial relation similarity between R_1 and R_2 is modeled as a function of $dx_{SPATIAL} = distance(dx_1, dx_2)$ and $dy_{SPATIAL} = distance(dy_1, dy_2)$.



R_1 = Spatial Relation between s_{11} and s_{12} : (dx_1, dy_1)
R_2 = Spatial Relation between s_{21} and s_{22} : (dx_2, dy_2)

Figure 30 Spatial Relations

4.5. Primitive Shape Feature Extraction

As mentioned earlier, the choice of features is often based on heuristics and it depends on the data. For each basic stroke, we would like to extract semantic information therefore we compare the basic stroke with some primitive shapes in order to understand how likely that stroke falls into a given primitive shape category. In [3], the geometry is used to recognize strokes into some primitive shapes. The geometry information is easily accessible since we are dealing with sketches in the stroke domain and not in the pixel domain as the case for images. Different geometric features are used to determine the likelihood that each stroke falls in each primitive shape: line, circle and polygon. Some examples of these features are illustrated in Figure 31.

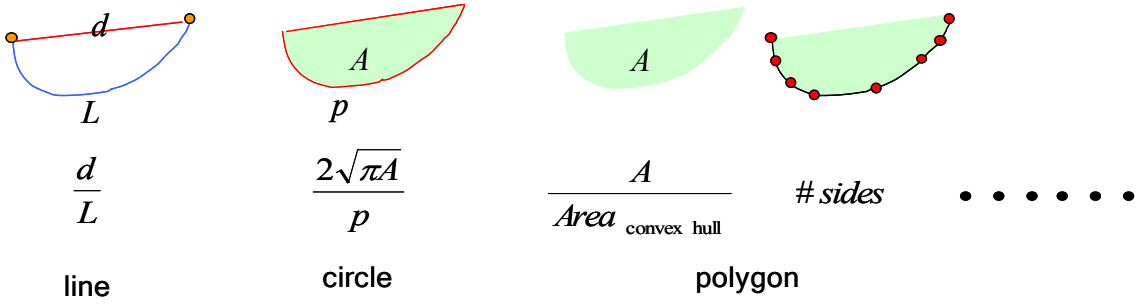


Figure 31 Example primitive shape features

Assume that a stroke with N samples is denoted by the sequence of 2-D coordinates (x_i,y_i), $i = 1,2,\ldots,N$. We would like to calculate the shape likelihood and the shape features for each stroke. Essentially the shape likelihood is used to describe the general characteristics of a primitive shape and the shape features are used to distinguish between the strokes that belong to similar primitive shape. The features that are used to calculate the shape likelihood and the features that are used in matching for each primitive shape are described in the following subsections.

4.5.1. Line Likelihood and Features

The features that are used for calculating the line likelihood are given as follows:

1) Average inverse height ratio

$$L_{line}^{1} = \max\left(0, 1 - \sum_{i=2}^{N-1} \frac{h_i}{d}\right) \tag{5}$$

$$\text{where } h_i = \frac{2A_i}{d} \tag{6}$$

$$d = \sqrt{(x_N - x_1)^2 + (y_N - y_1)^2} \tag{7}$$

$$A_i = \frac{1}{2}(x_1 y_i - x_i y_1 + x_i y_N - x_N y_i + x_N y_1 - x_1 y_N) \tag{8}$$

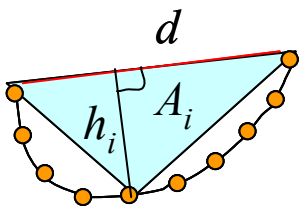


Figure 32 Illustration of average inverse height ratio

As illustrated in Figure 32, in each triangle formed by the end points and each of the non-end point samples, the height h_i is first computed by dividing the area of that triangle A_i by the distance d between the end points. The height ratio is obtained by further dividing h_i by d. The height ratio is averaged over all non-end point samples and then the inverse is calculated. If the result is less than 0, then this value is replaced with 0. Essentially this is a measure for the line primitive shape because if the stroke is very close to a line, then h_i is very small compared with d and the average inverse height ratio will thus be large.

2) Ratio between distance between end points and stroke length

$$L_{line}^2 = \frac{d}{L} \tag{9}$$

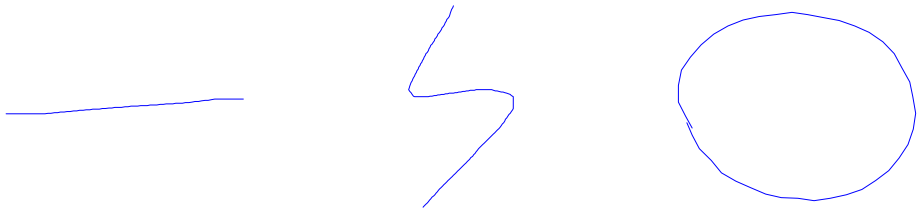
where d is given by equation (7) and L is given by equation (12). This ratio is very close to 1 if the stroke is very close to a line.

The overall line likelihood is obtained by the following equation (10)

$$L_{line} = \left(L_{line}^1\right)^{\lambda_{line}^1} \bullet \left(L_{line}^2\right)^{\lambda_{line}^2} \tag{10}$$

where λ_{line}^1 and λ_{line}^2 are fixed scalar weights.

Figure 33 shows the line likelihood of some stroke examples.



(a) $L_{line} = 0.9953$ (b) $L_{line} = 0.4958$ (c) $L_{line} = 0$

Figure 33 Line likelihood values of some strokes

The features for the line primitive shape that are used for matching are given as follows:

1) Estimated Slope

$$\alpha = \tan^{-1}\left(\frac{y_N - y_1}{x_N - x_1}\right) \tag{11}$$

This is exactly equal to the slope if the stroke is a line.

2) Stroke length

$$L = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{12}$$

4.5.2. Polygon Likelihood and Features

The features that are used for calculating the polygon likelihood are given as follows:

1) Area ratio between the stroke area and the area of its convex hull

$$L^1_{polygon} = \frac{\text{area of the convex hull}}{A} \tag{13}$$

$$\text{where } A = \frac{1}{2}\left[\sum_{i=1}^{N-1}\left(x_i y_{i+1} - x_{i+1} y_i\right) + \left(x_N y_1 - x_1 y_N\right)\right] \tag{14}$$

This measure is large if the stroke is close to a convex polygon.

2) Inverse percentage of number of stroke samples in the convex hull and the original number of stroke samples

$$L^2_{polygon} = 1 - \frac{\#\,\text{samples in the convex hull}}{N} \tag{15}$$

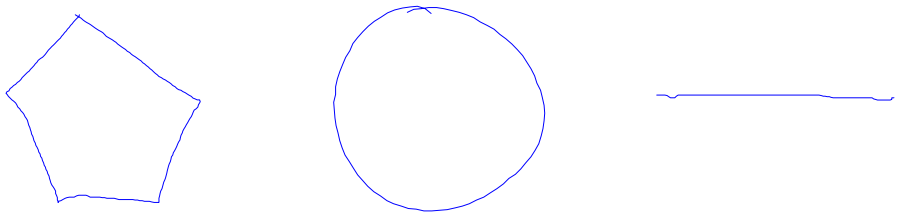
A polygon requires a relatively few number of points (ideally equal to the number of vertices) to form the convex hull that surrounds all the stroke samples. As a result, this inverse percentage is large if the stroke is close to a convex polygon.

The overall polygon likelihood is obtained by the following equation (16):

$$L_{polygon} = \left(L^1_{polygon}\right)^{\lambda^1_{polygon}} \bullet \left(L^2_{polygon}\right)^{\lambda^2_{polygon}} \tag{16}$$

where $\lambda^1_{polygon}$ and $\lambda^2_{polygon}$ are fixed scalar weights.

Figure 34 shows the polygon likelihood of some stroke examples.



(a) $L_{polygon} = 0.9772$ (b) $L_{polygon} = 0.5390$ (c) $L_{polygon} = 0$

Figure 34 Polygon likelihood of some strokes

The features for the polygon primitive shape that are used for matching are given as follows:

1) Number of sides

First consider the turn angle that is the amount of angle change at each sample point. As a result, the turn angle is 0 along a line since the angle does not change. The turn angle is computed by the following equation (17):

$$\beta_i = \cos^{-1} \frac{(x_{i+1} - x_i)(x_i - x_{i-1}) + (y_{i+1} - y_i)(y_i - y_{i-1})}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}} \tag{17}$$

The number of sides is estimated by first removing those sample points (x_i,y_i) that have small turn angles and then counting how many stroke segments are left with remaining stroke samples.

2) Area ratio between the stroke area and the area of its convex hull

This feature is exactly the same measure that is used for computing part of the polygon likelihood given by equation (13).

3) Closeness

$$c = \frac{L}{p} \tag{18}$$

$$\text{where } p = L + d \tag{19}$$

L is the stroke length given by equation (12) and d is the distance between end points given by equation (7). Therefore p is the perimeter of shape formed by the stroke. Essentially this closeness measures how close the end points are since when the end points are close, d is very small and c is close to 1.

4) Perimeter efficiency

$$k = \frac{2\sqrt{\pi A}}{p} \tag{20}$$

This feature measures much the shape is close to a circle since $k = 1$ implies that the shape is a circle. In addition, this feature can also be used to distinguish between polygons since a polygon with more sides tend to have a higher value of k since it is more similar to a circle. For example, for a regular triangle, $k = 0.79$; for a regular square, $k = 0.89$; for a regular pentagon, $k = 0.93$.

4.5.3. Circle Likelihood and Features

The features that are used for calculating the polygon likelihood are given as follows:

1) Perimeter efficiency

$$L^1_{circle} = k \tag{21}$$

k is given by equation (20). As mentioned before, $k = 1$ implies that the shape is a circle.

2) Average of estimated radius

$$L^2_{circle} = \frac{\frac{1}{\lfloor N/3 \rfloor}\sum_{j=1}^{\lfloor N/3 \rfloor} r_j}{\max_j (r_j)} \tag{22}$$

where r_j is the estimated radius of a circle that can be constructed from three stroke sample points. We divide the stroke samples into 3 sets and each time we pick one sample from each set in order to estimate the radius. As illustrated in Figure 35,

given three sample stroke points (x_{j1},y_{j1}), (x_{j2},y_{j2}) and (x_{j3},y_{j3}), we first find out where is the center by constructing perpendicular bisector for the line segment from (x_{j1},y_{j1}) to (x_{j2},y_{j2}) and from (x_{j2},y_{j2}) to (x_{j3},y_{j3}). Then the radius can be computed by calculating the distance between the center to any of the stroke sample point.

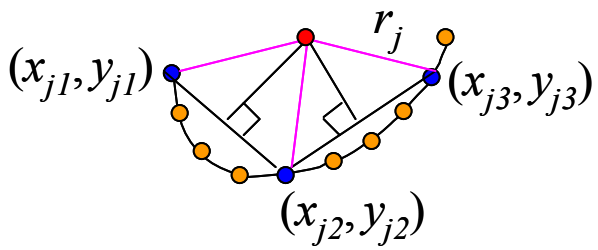


Figure 35 Illustration of estimated radius

3) Total turn angle

$$L^3_{circle} = \begin{cases} \dfrac{1}{2\pi}\sum_{i=2}^{N-1}\beta_i, & \sum_{i=2}^{N-1}\beta_i <= 2\pi \\ \dfrac{2\pi}{\sum_{i=2}^{N-1}\beta_i}, & \sum_{i=2}^{N-1}\beta_i > 2\pi \end{cases} \tag{23}$$

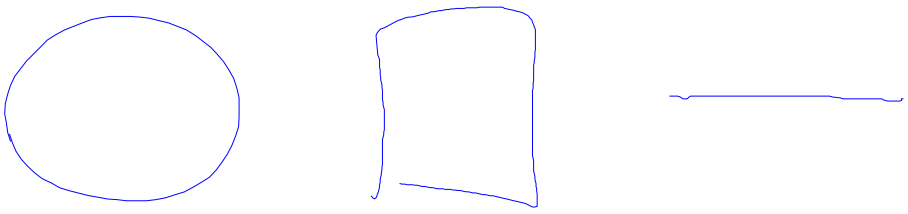
where β_i is the turn angle given by equation (17). The total turn angle is 2π for a circle. This measure increases as the total turn angle goes from 0 to 2π, reaches the maximum at 2π, and then decreases as the total turn angle continues to increase.

The overall polygon likelihood is obtained by the following equation (16):

$$L_{circle} = \left(L^1_{circle}\right)^{\lambda^1_{circle}} \bullet \left(L^2_{circle}\right)^{\lambda^2_{circle}} \bullet \left(L^3_{circle}\right)^{\lambda^3_{circle}} \tag{24}$$

where λ^1_{circle} , λ^2_{circle} and λ^3_{circle} are fixed scalar weights.

Figure 36 shows the circle likelihood of some stroke examples.



(a) $L_{circle} = 0.8536$ (b) $L_{circle} = 0.6806$ (c) $L_{circle} = 0.273$

Figure 36 Circle likelihood values of some strokes

The features for the circle primitive shape that are used for matching are given as follows:

1) Area ratio between the stroke area and the area of its convex hull

   This is the same measure as the one given in equation (13).

2) Total turn angle

   This is similar to the one given in equation (23) where β_i is still the turn angle given by equation (17).

$$\gamma = \frac{1}{2\pi} \sum_{i=2}^{N-1} \beta_i \tag{25}$$

3) Perimeter efficiency

   This is the same feature as given in equation (20).

4.5.4. Non-Primitive Shape Likelihood and Features

When the stroke does not look like any of the primitive shapes, we have another likelihood and another set of features to take care of this case. The non-primitive shape likelihood is given by equation (26).

$$L_{none} = \left(1 - L_{line}\right)^{\mu_{line}} \bullet \left(1 - L_{polygon}\right)^{\mu_{polygon}} \bullet \left(1 - L_{circle}\right)^{\mu_{circle}} \tag{26}$$

where μ_{line} , $\mu_{polygon}$ and μ_{circle} are fixed scalar weights.

The features for the non-primitive shape that are used for matching are given as follows. These features are chosen because they provide a more general description about a shape as we do not have specific knowledge about a non-primitive shape.

1) Stroke area

   This is the same feature as given in equation (14).

2) Area ratio between the stroke area and the area of its convex hull

   This feature is exactly the same measure that is used for computing part of the polygon likelihood given by equation (13).

3) Closeness

   This is the same feature as given in equation (18).

4.5.5. Heuristic Scalar Weights

All the scalar weights are determined by heuristics and they are provided in the following table:

Shape Type	Weight	Value
Line	λ^1_{line}	1.0
	λ^2_{line}	1.0
Polygon	$\lambda^1_{polygon}$	0.10
	$\lambda^2_{polygon}$	0.45
Circle	λ^1_{circle}	0.33
	λ^2_{circle}	1.0

	λ^3_{circle}	0.33
Non-primitive Shape	μ_{line}	0.25
	$\mu_{polygon}$	0.25
	μ_{circle}	0.25

Table 2 Heuristic Scalar Weights

4.6. Appendices

4.6.1. Appendix A: Simplified Shade Detection for Sketches

As mentioned in Section 4.1, a region is more likely to be a shaded region if the ink density of the strokes in that area is high. Further, if we assume that the user always draw a boundary enclosing the shaded region, then we can use the following method to perform shade detection. In order to determine whether a region has high ink density, we consider the total stroke length (the sum of the length of all the descendent strokes) and the convex hull area of the parent stroke. We plot these features from a set of training data in Figure 37, where the shaded region is characterized by large total stroke length with small convex hull area. The training data consists of shaded and non-shaded regions with both large and small areas as shown in Figure 37. The decision boundary is selected as follows:

$$x \begin{array}{c} \text{shaded} \\ > \\ < \\ \text{non-shaded} \end{array} Ay^2 + By + C \tag{27}$$

where x is the stroke length; y is the convex hull area; A, B and C are parameters determined from the training data. A second order function is chosen as the decision boundary because the training data cannot be well separated by a linear function.

With this approach, the strokes forming the shaded region can still be considered as a hyper-stroke since under the assumption, the user draws a boundary that can be

considered as the basic stroke and there exist other strokes inside the boundary forming the shade.

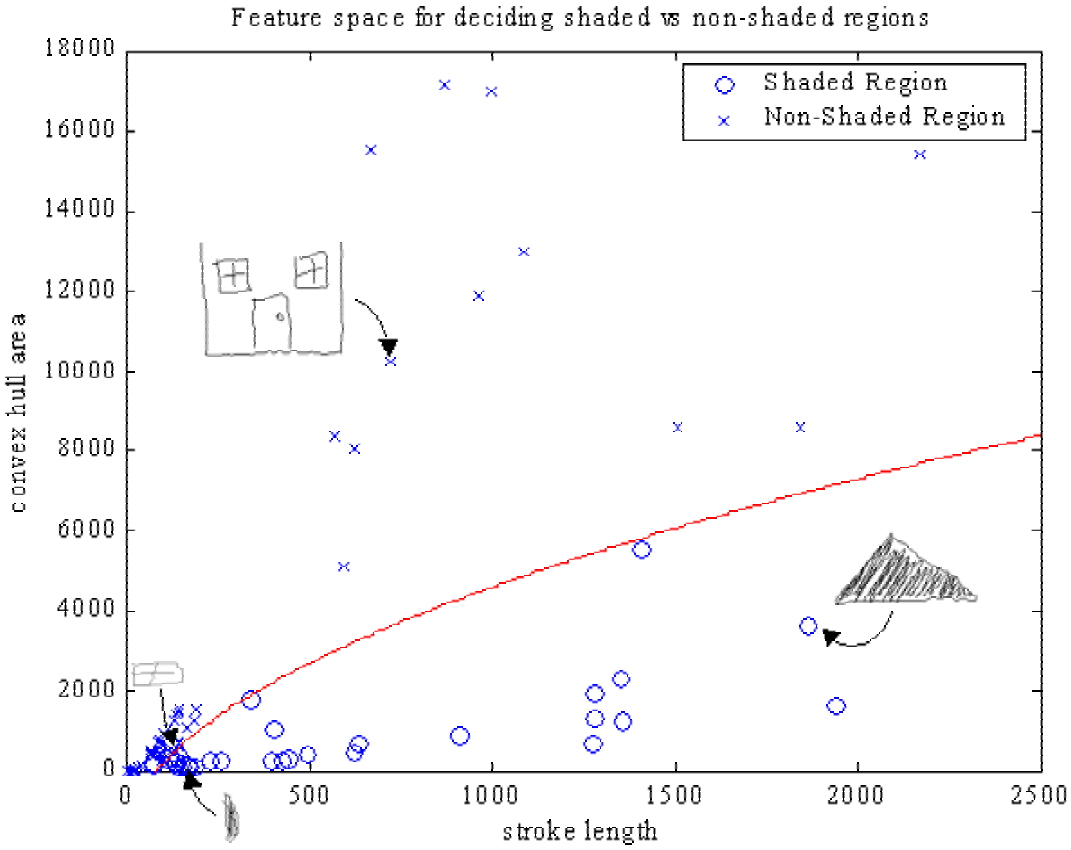
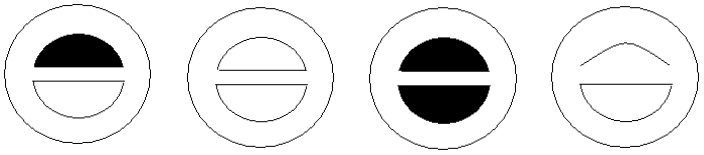


Figure 37 Feature space for deciding shaded vs. non-shaded region

4.6.2. Appendix B: Shade Detection For Images

The idea of shade detection in sketches can also be applied for images because sometimes an image can be described a combination of solid regions and edges. For each region, thinning or edge extraction may be applied since one method is preferred to the other under different situations. For example, if edge extraction alone is used to extract the contour, then both the images in Figure 38(b) and Figure 38(c) are considered to be the same as the image in Figure 38(a) although they have different solid regions. If

thinning alone is used to extract the skeleton, then the image in Figure 38(d) is considered to be very similar to the image in Figure 38(a). This gives us the motivation of classifying a region with contour vs. skeleton representation. After the contour-skeleton classification, stroke tracing is performed to extract the sketch. The user can provide a query sketch that will be compared with those extracted sketches from the database trademark images in order to retrieve similar images.



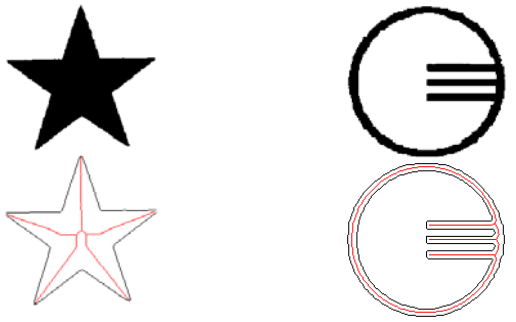
(a) (b) (c) (d)

Figure 38 Example images with and without solid regions

For each region, either edge extraction is performed to extract the contour or thinning is performed to extract the skeleton. It is advantageous to use different methods under different situations. For example, for a solid region in which the shape conveys a lot of visual information, it is better to perform edge extraction to that region to extract the contour. On the other hand, for a region that contains curves, thinning should be performed to that region to extract the skeleton that is a better representation.

To determine whether contour or skeleton is a better representation for a region, we compute the *thickness* that is the distance between each pixel of the skeleton and the nearest pixel of the contour. For example, we would like to perform thinning for a region if the thickness is small and if it does not vary too much across different skeleton pixel values as shown in Figure 39(b). On the other hand, if there is a large variation in the thickness, then edge extraction is preferred to extract the contour as shown in Figure 39(a). As a result, the mean and variance of the thickness is used for classifying a region

into skeleton-best or contour-best representation. After performing edge extraction or thinning for all the regions, the strokes are traced by examining the pixel connectivity starting from the end points. This results in two types of strokes: contour strokes which are obtained by edge extraction and skeleton strokes which are obtained by thinning.



(a) Region that is suitable for edge extraction

(b) Region that is suitable for thinning

Figure 39 Example regions that are suitable for edge extraction and for thinning, and their corresponding skeleton superimposed on the contour

We analyze the performance of our classifier for deciding whether a contour or a skeleton should be extracted for each region. First we need to train the classifier to obtain the decision boundary. As mentioned in the previous section, the mean and variance of the thickness are used as the input features for the classifier. We compute these features for about 600 ground truth trademark images that we know which are good for contour stroke representation and which are good for skeleton stroke representation. The resulting distribution of the mean and variance of the thickness are shown in Figure 40. The decision boundary is based on the Mahalanobis distances from the input feature vector to the mean feature of each of the two classes. Assume that x is the input feature vector, i.e., x = [mean(thickness) var(thickness)]^T, m_s and Σ_s represent the mean and the covariance matrix of the feature vector for the trademark images that are best represented by skeleton strokes and m_c and Σ_c represent the mean and the covariance matrix of the

feature vector for the trademark images that are best represented by contour strokes. Then the classification criterion is given as follows:

$$-(x-m_s)^T \Sigma_s^{-1}(x-m_s) - \ln(\det(\Sigma_s)) \underset{contour}{\overset{skeleton}{\gtrless}} -(x-m_c)^T \Sigma_c^{-1}(x-m_c) - \ln(\det(\Sigma_c)) \tag{28}$$

The resulting decision boundary is shown in Figure 40.

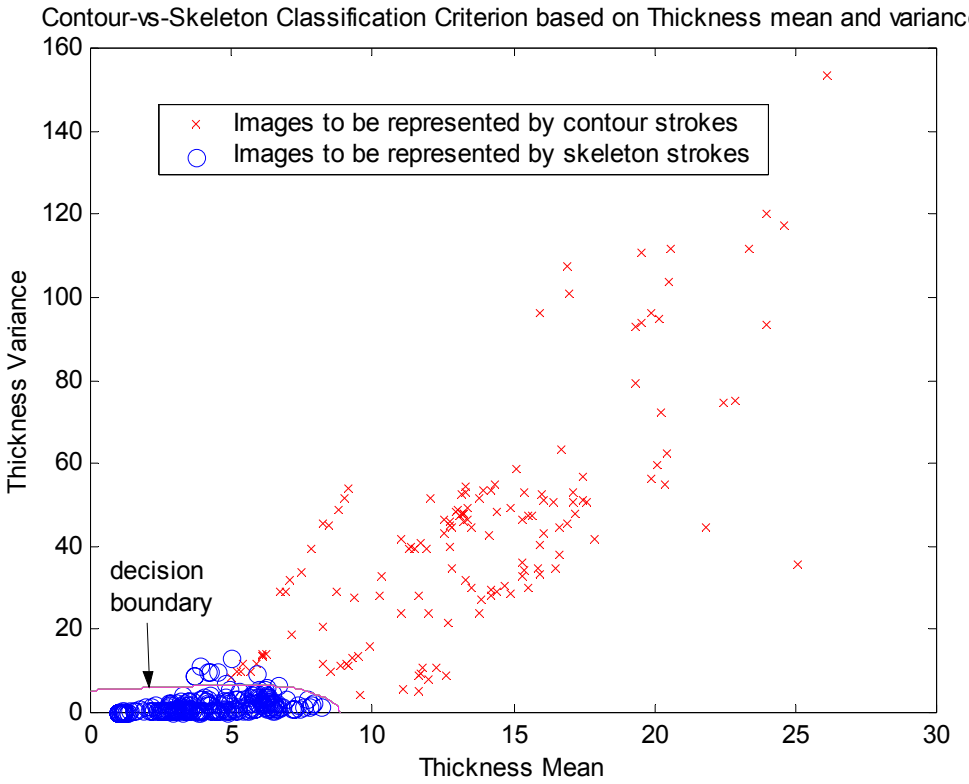


Figure 40 Contour-skeleton classification criterion

5. Global And Local Matching

This chapter describes the global and local matching of features at different levels. While a sketch consists of multiple components and two sketches may have different number of components, it is necessary to first find a correspondence between the components in the two sketches. This correspondence is determined from global matching by minimizing the total cost (or maximizing the total similarity score) between components. Based on this correspondence, the similarity score between the components can be computed. On the other hand, local matching that compares attributes of only a few components independently can also be performed such as the matching of stroke hierarchy and the spatial relations.

This chapter is organized as follows: Section 5.1 introduces the multiple component feature correspondence problem and discuses two ways of approaching the problem. Section 5.2 provides the similarity score computation for matching different levels of features in order to compare both global and local information. Section 5.3 presents the experimental result by comparing the proposed approach with two other approaches.

5.1. Multiple Component Feature Correspondence

A sketch consists of multiple components therefore in the feature space, it can be represented by a set of feature points. The number of components in a sketch depends on its complexity hence two sketches may contain different number of components. As a

result, when two sketches are being matched, we need to first determine the correspondence between the components before the similarity score can be computed. As illustrated in Figure 41, assuming that we know the cost matrix whose entry is the matching cost between each component of sketch 1 and each component of sketch 2, determining the correspondence between the components is to find a matching such that the total cost is minimized. In graph theory, this corresponds to the bipartite graph matching problem [5][10]. It can also be considered as the marriage problem where there are *M* men and *M* women, and the goal is to find the best way of arranging the *M* couples based on their scores on each candidate. This problem can be solved by the Hungarian method [21]. In our sketch retrieval problem, we need to find a set of correspondence between *M* feature points from one sketch with *N* feature points from another sketch. The cost matrix can be constructed by considering the inverse of the similarity between each pair of feature points. However, since the Hungarian method requires the cost matrix to be square, some dummy rows or columns need to be appended to the cost matrix.

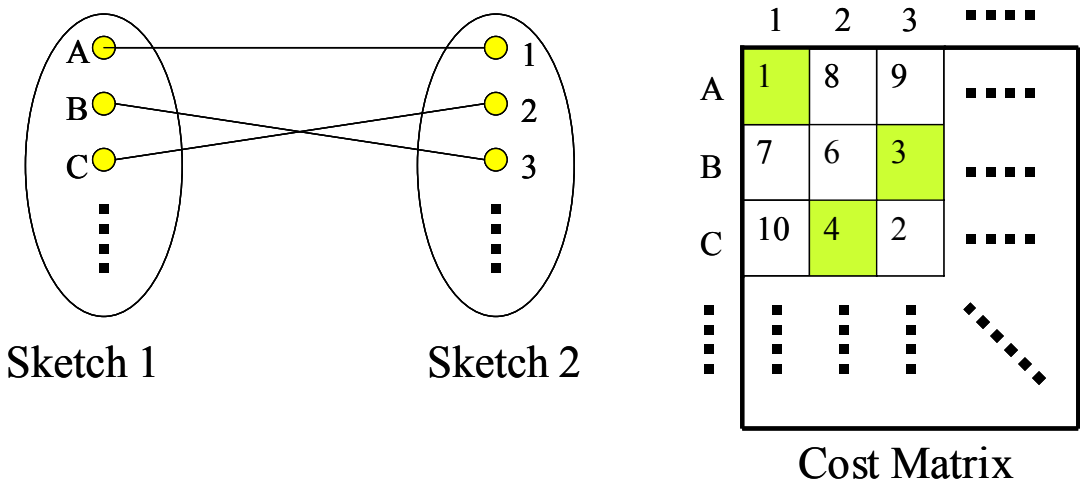


Figure 41 Matching between multiple components given the cost matrix

From the Hungarian method, the correspondence between multiple strokes can be assigned according to the shape information between strokes. In this case, in Figure 42, the triangles in sketch 2 and sketch 3 are corresponding strokes to the triangle in sketch 1 and the rectangles in these two sketches are corresponding strokes to the rectangle in sketch 1. If the similarity is entirely based on shape, then sketch 2 and sketch 3 are both similar to sketch 1. On the other hand, if the spatial relation between the strokes is also considered, then we can make a further distinction that sketch 2 is more similar than sketch 3 with respect to sketch 1. There are two ways of including the spatial relations into consideration: 1) use the spatial relations in addition to shape information when determining the stroke correspondence; or 2) keep using only shape information when determining the stroke correspondence, but includes the spatial relations in the similarity computation after the correspondence is found.

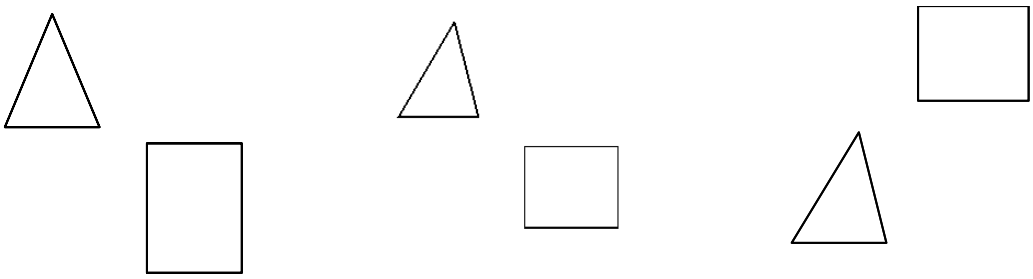


Figure 42 Example sketches to demonstrate spatial relations

In the first way, the spatial relations are also considered in determining the stroke correspondence. There exist some methods [38][43] that try to achieve this goal by matching shape and spatial relations at the same time. Petrakis and Faloutsos [38] used attributed relational graphs (ARG) for matching medical images under the assumption that the number of feature points of the query is less than or equal to the number of

feature points of a database sketch. Smith and Chang [43] integrated the spatial and shape features by using the 2D string matching technique proposed in [8] that maintains a consistent order for the *x*-projection and the *y*-projection of all matched components. Assume that the orders of the strokes are preserved in determining the correspondence. In Figure 42, in the horizontal direction, the stroke correspondence of the sketches is the same as the previous case when only shape information is considered. However, in the vertical direction, only one of the object pairs (the triangle pair and the rectangle pair) can be corresponding strokes for sketch 2 and sketch 3 with respect to sketch 1 but not both because the two object pairs have conflicting orders in the vertical direction. The object pair to be selected as the corresponding strokes depends on the shape similarity of the object pairs. For example, if the triangle pair has a larger similarity score than the rectangle pair, then the triangles will be the only corresponding strokes in the vertical direction. The horizontal and vertical directions are considered separately for the stroke correspondence and then the similarity score is the combined result in both directions.

In the second way, the correspondence is determined solely based on shape information, and then spatial relations are used in the similarity computation. In [35][36], the spatial relation between multiple objects is used for similarity computation assuming that the object correspondence is given. In this case, when considering the example in Figure 42, the stroke correspondence between sketch 1 and sketch 2 is the same as that between sketch 1 and sketch 3. However, if we include spatial relations in the similarity computation, then the similarity score between sketch 1 and sketch 2 will still be higher than that between sketch 1 and sketch 3 since sketch 1 and sketch 2 are more similar in terms of spatial relations.

The first way has the advantage that shape and spatial relations are considered simultaneously therefore the matched result can be more accurate under full search. However, the stroke correspondence problem becomes much more complex when trying to match both shape and spatial relations simultaneously thus full search is not feasible. It is thus necessary to reduce the search space by limiting the search paths. As a result, error can still be introduced under the first way. On the other hand, in the second way, the stroke correspondence is a much simpler problem since only shape information is considered and the Hungarian method provides an efficient way of solving this problem. Assume that there are not too many similar sets of strokes within the same sketch such that there are not too many mistakes in the stroke correspondence, then the spatial relations are helpful in computing the similarity score to adjust the ranks of the retrieved result properly. As a result, we propose to use the second way to determine the multiple component feature correspondence by considering only the shape similarity and then include spatial relations in the similarity computation.

5.2. Similarity Functions

We introduce the similarity functions that we use for matching different levels of features in order to compare both global and local information.

5.2.1. Stroke Hierarchy Similarity

The similarity in the stroke hierarchical structures is determined by counting how many corresponding stroke pairs also preserve the parent-child relationship in the stroke hierarchies. For example, two stroke hierarchies are shown in Figure 43 where nodes with the same numbers are corresponding strokes. Between these two stroke hierarchies,

three corresponding stroke pairs (2-5; 2-6 and 3-4) are also parent-child strokes in both hierarchies. As a result, the similarity in the stroke hierarchical structure in this case is 3.

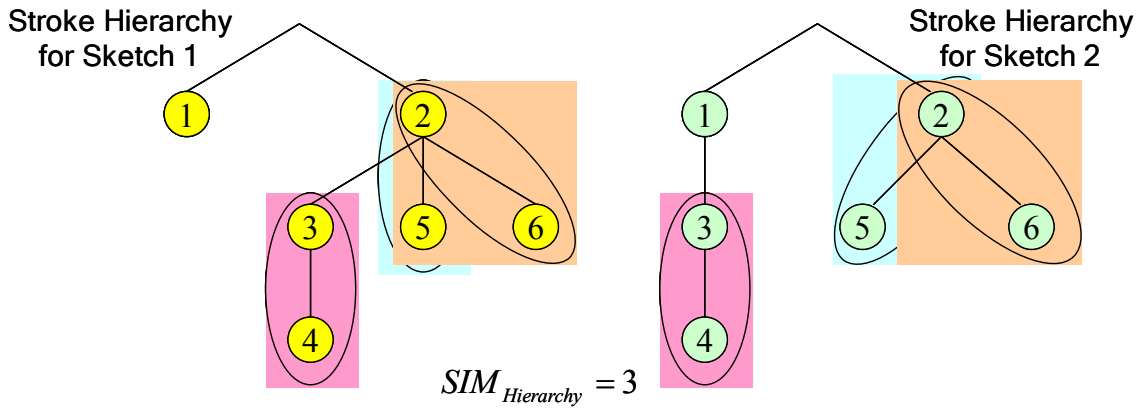


Figure 43 Stroke hierarchy similarity

5.2.2. Hyper-Stroke Similarity

We first show how to compute the similarity between two hyper-strokes. Then we will describe how to compute the similarity between two sets of hyper-strokes where each set of hyper-strokes is generated from one sketch.

We use the Mahalanobis distance between the hyper-stroke features F_1 and F_2 as the feature distance measure as shown in equation (29)

$$d_{HYPER}(F_1, F_2) = (F_1 - F_2)^T \Sigma^{-1} (F_1 - F_2) \quad (29)$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the hyper-stroke features pre-determined from the sketches in the database.

The hyper-stroke similarity between two hyper-strokes is computed as the inverse of the hyper-stroke feature distance.

A set of hyper-strokes can be generated from one sketch. As a result, given two sketches, we need to compare two sets of hyper-strokes by first determining the

correspondence so that we know which hyper-stroke from one sketch is matched with which hyper-stroke from another sketch. In determining the correspondence, first the similarity table is constructed where each element represents the similarity between a hyper-stroke from sketch 1 and a hyper-stroke from sketch 2. Then the correspondence between the two sets of hyper-strokes is determined by the greedy algorithm. The element in the table with the maximum score is selected and the associated row index and column index of that element indicates a pair of corresponding hyper-strokes. The row and the column associated with that element are removed from the table. The maximum score is searched again from the remaining table and this process is repeated until the table becomes empty. The resulting similarity score between the two sets of hyper-strokes is computed as the sum of the similarity scores of the selected elements.

5.2.3. Spatial Relation Similarity

Recall in Section 4.4, the spatial relation is defined by the displacement vector between the centers of two strokes within a sketch. Using the notations illustrated in Figure 30, the spatial relation similarity between R_1 and R_2 is modeled as a function of $dx_{SPATIAL} = distance(dx_1, dx_2)$ and $dy_{SPATIAL} = distance(dy_1, dy_2)$ defined by equation (30).

$$Sim(R_1, R_2) = e^{-|dx_{SPATIAL}|} + e^{-|dy_{SPATIAL}|} \tag{30}$$

An exponential function is used as the spatial relation similarity. Since it is a continuous function, there will not be a big jump in the spatial relation similarity when the location of a stroke is changed a little bit as opposed to the case of using the spatial ordering or the interval relationship by quantizing the spatial relations into discrete steps.

5.2.4. Shape Similarity

As described in Section 4.5, there are features for three primitive shapes and a non-primitive shape for each stroke. The shape similarity is computed by first calculating the similarity scores across all shapes and then picking the maximum score.

In [45], equation (31) is proposed to be used as the similarity function. We also use equation (31) to match two features F_{i1j} and F_{i2j}, which are the j-th feature of the i-th shape (3 primitive shapes + 1 non-primitive shape) for strokes s_1 and s_2.

$$Sim(s_{i1j}, s_{i2j}) = \frac{\min(F_{i1j}, F_{i2j})}{\max(F_{i1j}, F_{i2j})} \tag{31}$$

If the feature spans a large range, then it is not suitable to use equation (31) as the similarity function any more since the decay may be too fast as the feature distance increases. As a result, we propose another similarity function as shown in equation (32) to compare features that span a large range. This is essentially the ratio between the geometric mean and the arithmetic mean thus the score lies between 0 and 1.

$$Sim(s_{i1j}, s_{i2j}) = \frac{\sqrt{(F_{i1j} \bullet F_{i2j})}}{(F_{i1j} + F_{i2j})/2} \tag{32}$$

The resulting shape similarity score between two strokes s_1 and s_2 is given by equation (33).

$$SIM_{SHAPE}(s_1, s_2) = \max_i \left[L_i(s_1) L_i(s_2) \prod_j Sim(s_{i1j}, s_{i2j}) \right] \tag{33}$$

5.2.5. Overall Similarity

Figure 44 provides a unified system by how the representation stage, the feature extraction stage and the matching stage are combined. The numbers in the brackets

denote which section the underlying blocks are described. This provides a summary of how Chapter 3, Chapter 4 and Chapter 5 are bonded together to form the framework.

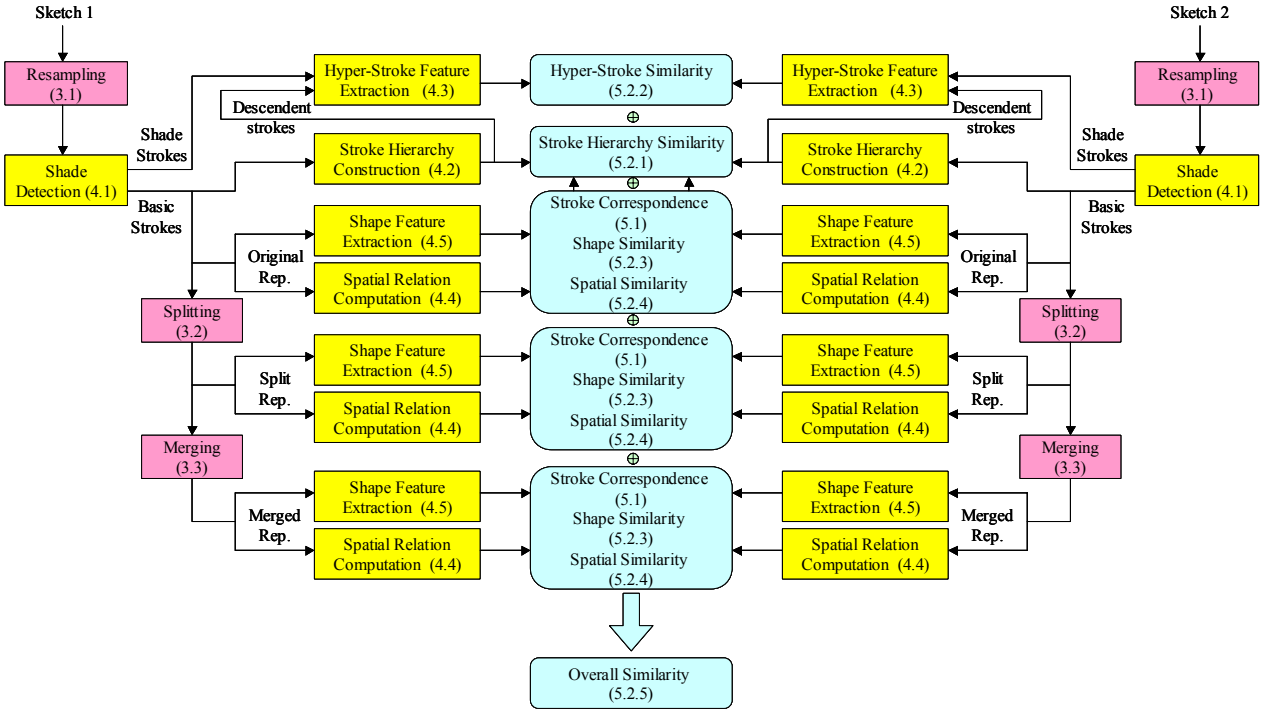


Figure 44 Unified system with representation, feature extraction and matching

Now we provide a description about how to compute the similarity scores for different levels of features between two sketches. The blocks in the center column in Figure 44 corresponds to the matching stage. The hyper-stroke similarity between the two sets of hyper-stroke features from sketch 1 and sketch 2 is computed according to Section 5.2.2. The stroke hierarchies from the two sketches are also compared according to Section 5.2.1. Then for each of the multiform representations (original, split, merged), the similarity of the basic strokes between the two sketches is computed. This is done by first determining the correspondence between the basic strokes with the Hungarian method specified in Section 5.1. The shape similarity between each corresponding stroke pair between two sketches is computed according to 5.2.4. The spatial relation similarity

is computed between a pair of strokes in sketch 1 and the pair of corresponding strokes in sketch 2 according to Section 5.2.3. The similarity between the two sketches in each multi-form representation is then computed by combining the shape and spatial similarity functions as follows:

$$SIM_{SHAPE+SPATIAL}(\text{Sketch 1}, \text{Sketch 2}) = \sum_{p=1}^{n-1} \sum_{q=p+1}^{n} Sim(R_{1pq}, R_{2pq}) \left[Sim(s_{1p}, s_{2p}) + Sim(s_{1q}, s_{2q}) \right] \quad (34)$$

where s_{1p} and s_{2p} are the p-th corresponding strokes between sketch 1 and sketch 2, R_{1pq} and R_{2pq} are the spatial relations between the p-th and the q-th corresponding strokes in Sketch 1 and Sketch 2 respectively, and n is the number of corresponding strokes between the two sketches. This equation assumes that there are at least two strokes in each sketch. In case there is only a single stroke in one of the sketches, then only the shape similarity is considered without using the spatial relation similarity.

We now describe how to compute the overall similarity score from the similarity scores for different levels of features. For each of the three multi-form representations, the shape and spatial relation similarity scores between the query sketch and each sketch in the database are computed. Then the mean of these scores for each representation is calculated. Afterwards the shape and spatial similarity scores of each representation are normalized by dividing themselves by the mean score of that representation. The resulting scores are then combined linearly with certain weights that are initialized with equal values and can be updated with the relevance feedback method described in Section 6.2 . The final overall similarity score is obtained by another linear combination of the shape and spatial relation similarity of the multi-form representations with the hyper-stroke feature similarity and the stroke hierarchy similarity.

$$SIM_{OVERALL} = w_{SHAPE+SPATIAL} \bullet SIM_{SHAPE+SPATIAL} + w_{HYPER} \bullet SIM_{HYPER} + w_{HIERARCHY} \bullet SIM_{HIERARCHY} \quad (35)$$

5.3. Experiment and Result

Figure 45 shows the retrieval performance of our proposed approach compared with two other approaches. The weights that we used in this experiment are $w_{SHAPE+SPATIAL} = 0.9, w_{HYPER} = 0.05, w_{HIERARCHY} = 0.05$. The first approach uses the Hu moment invariants [15] as features and the second approach uses the wavelet coefficients [16]. It can be seen that by using our proposed approach, the result is better than the previous two approaches. We also compare our performance with the result obtained by using the linear combination of the two approaches. Although the performance improves after combining the two approaches, our approach still outperforms this combined result.

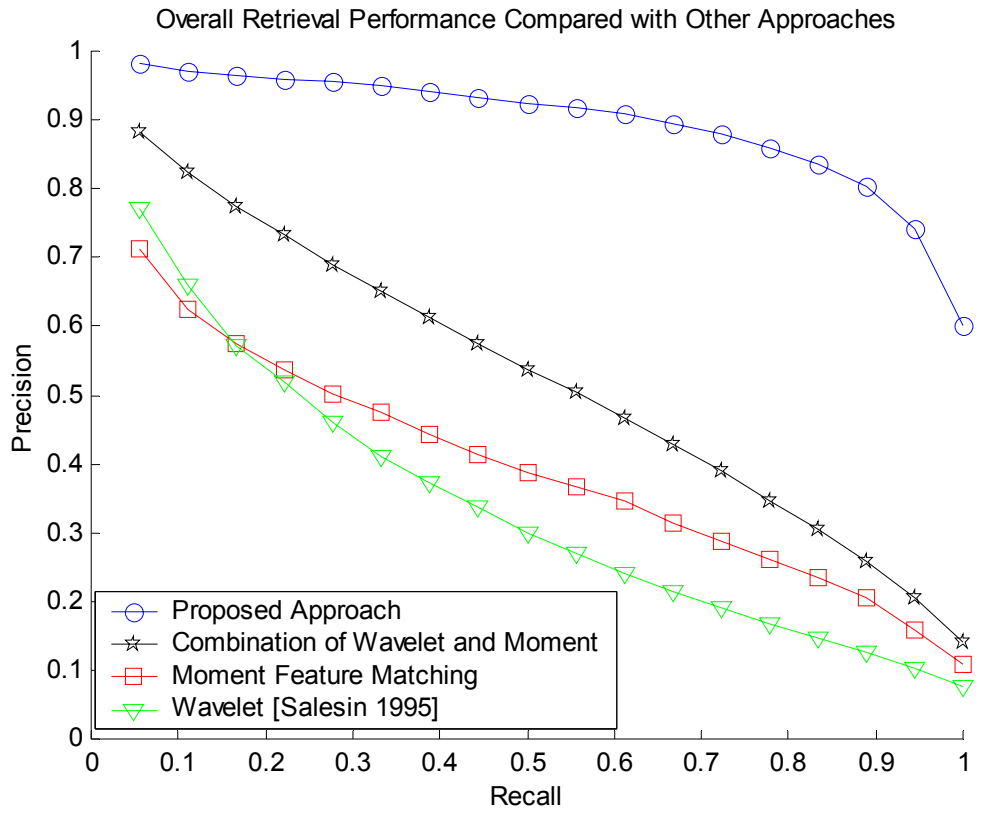


Figure 45 Comparison of retrieval performance with other approaches

Now we perform other experiments to show the gain in retrieval performance with respect to individual module. The gain in retrieval performance by combining the matching results from multiform representations has already been shown in Section 3.4. We perform another experiment to compare the retrieval performance with and without stroke hierarchy similarity. To make the dataset more challenging, we add two new classes of sketches whose examples are shown in Figure 46 and in Figure 47. The example sketches shown in Figure 46 illustrate the sign showing traffic light ahead whereas the example sketches shown in Figure 47 illustrate that there is a sign below the traffic light. It should be noted that these two classes of sketches contain strokes of the same shape, yet the fact that the traffic light is drawn inside or above the diamond shape gives two different meanings. As a result, we expect the stroke hierarchy to be able to help in the retrieval. In this setup, the retrieval performance using only the original representation is compared with the retrieval performance using the original representation with stroke hierarchy similarity. The overall similarity score used in the former case is obtained by setting $w_{SHAPE+SPATIAL} = 1, w_{HYPER} = 0, w_{HIERARCHY} = 0$ and the resulting similarity score used in the latter case is obtained by setting $w_{SHAPE+SPATIAL} = 0.5, w_{HYPER} = 0, w_{HIERARCHY} = 0.5$. Figure 48 shows the retrieval performance using the "Sign – traffic light ahead" sketches as the query. It can be seen that the performance is better when stroke hierarchy similarity is considered.

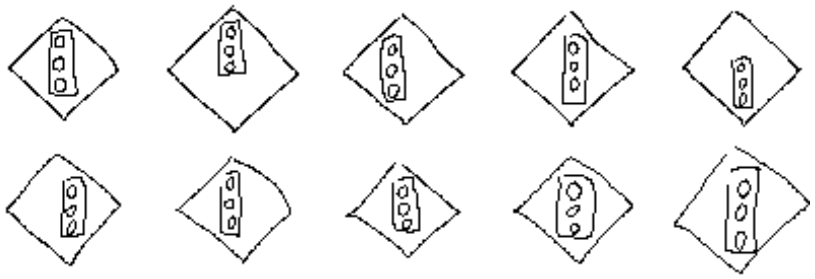


Figure 46 Example Sketches in the Class "Sign – traffic light ahead"

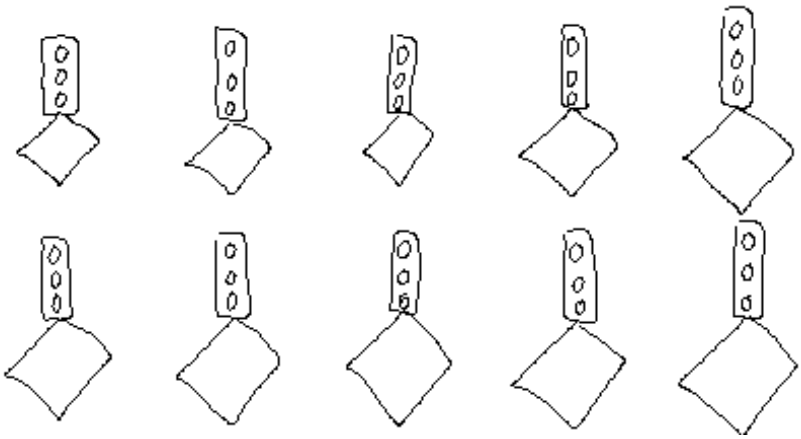


Figure 47 Example Sketches in the Class "Sign below traffic light"

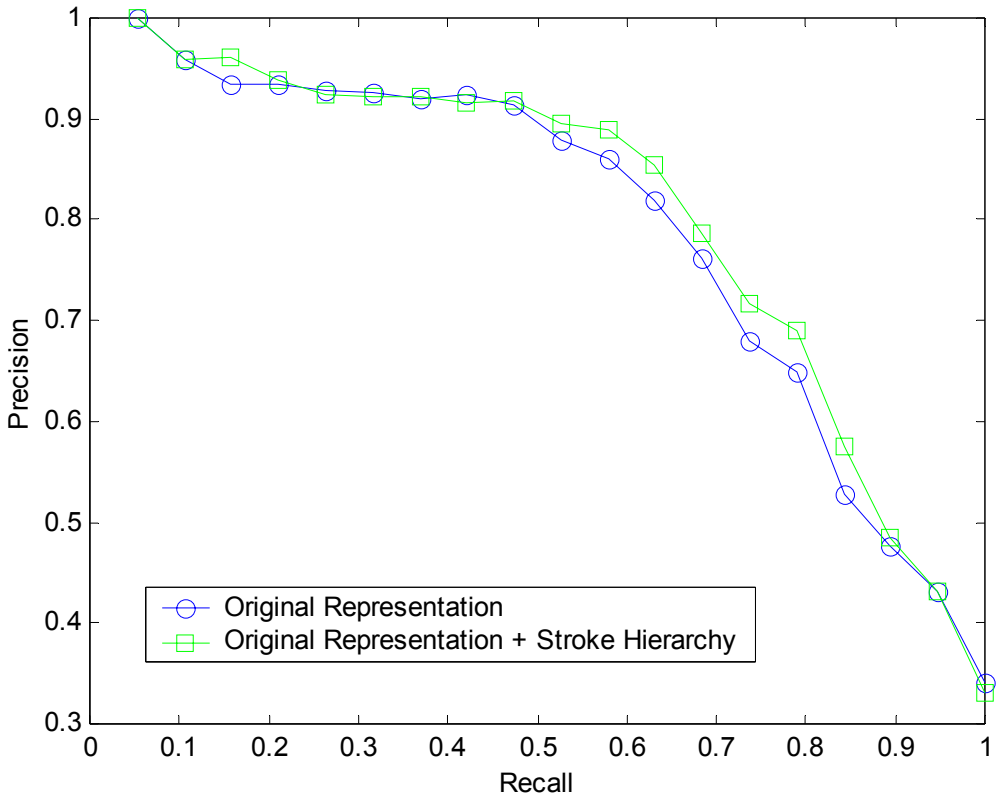


Figure 48 Comparison of retrieval performance for stroke hierarchy

We perform another experiment to compare the retrieval performance with and without hyper-stroke similarity. We use a new class of sketches shown in Figure 49 as the queries. These sketches illustrate different kinds of wall sockets but they are considered as one class. The challenge about this class of sketch is that the shapes between different kinds of wall sockets are quite different. However, all of them have the property that it has three holes that are indicated by the three shaded regions. As a result, we expect the hyper-stroke to be able to help in the retrieval. In this setup, the retrieval performance using only the original representation is compared with the retrieval performance using the original representation with hyper-stroke similarity. The overall similarity score used in the former case is obtained by setting $w_{SHAPE+SPATIAL} = 1, w_{HYPER} = 0, w_{HIERARCHY} = 0$ and the resulting similarity score used in the latter case is obtained by setting $w_{SHAPE+SPATIAL} = 0.5, w_{HYPER} = 0.5, w_{HIERARCHY} = 0.0$. Figure 50 shows the retrieval performance and it can be seen that the performance is better when hyper-stroke similarity is considered.

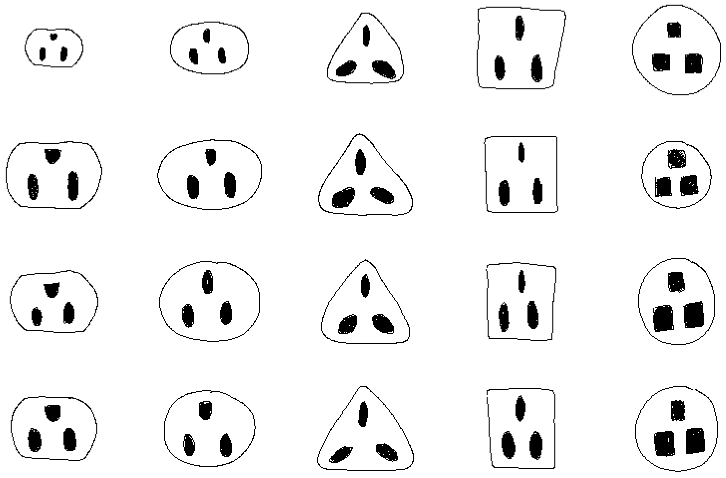


Figure 49 Sketches in the Class "Wall Socket"

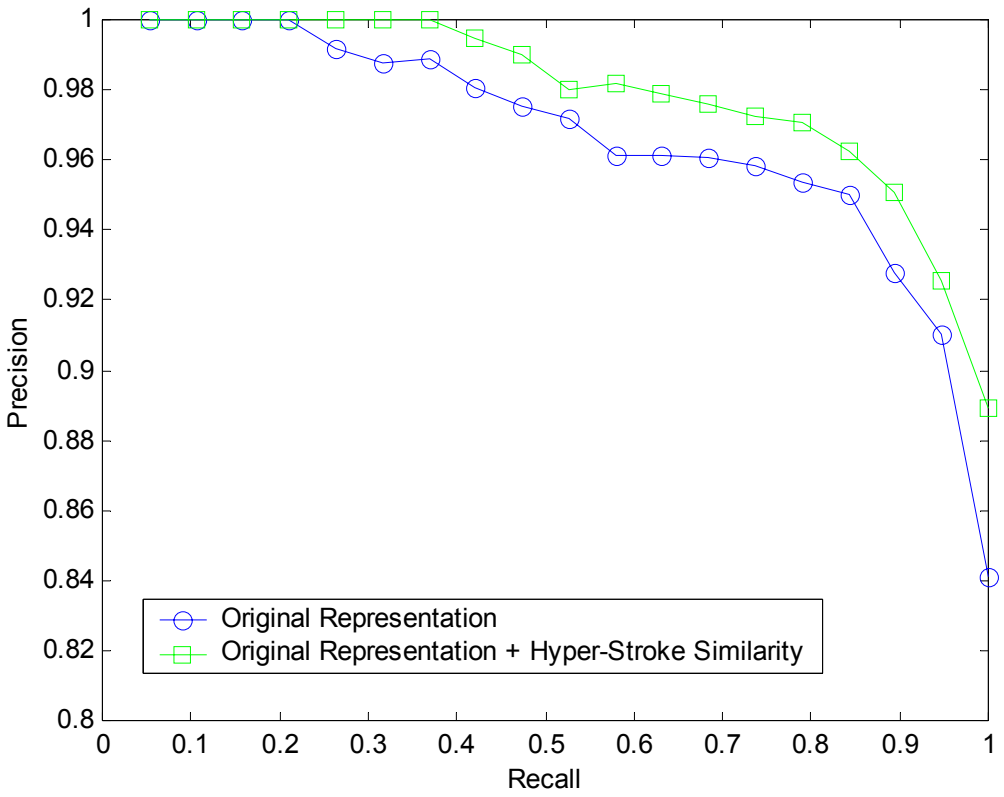


Figure 50 Comparison of retrieval performance for hyper-stroke similarity

6. Multiple Component Relevance Feedback

Sometimes the system may not provide satisfactory retrieval results after the user provides a query. Relevance feedback has been proposed to refine the retrieval by asking the user to provide positive and negative examples from the retrieval result. The system then learns from these examples and hopefully the new retrieval result is closer to what the user wants.

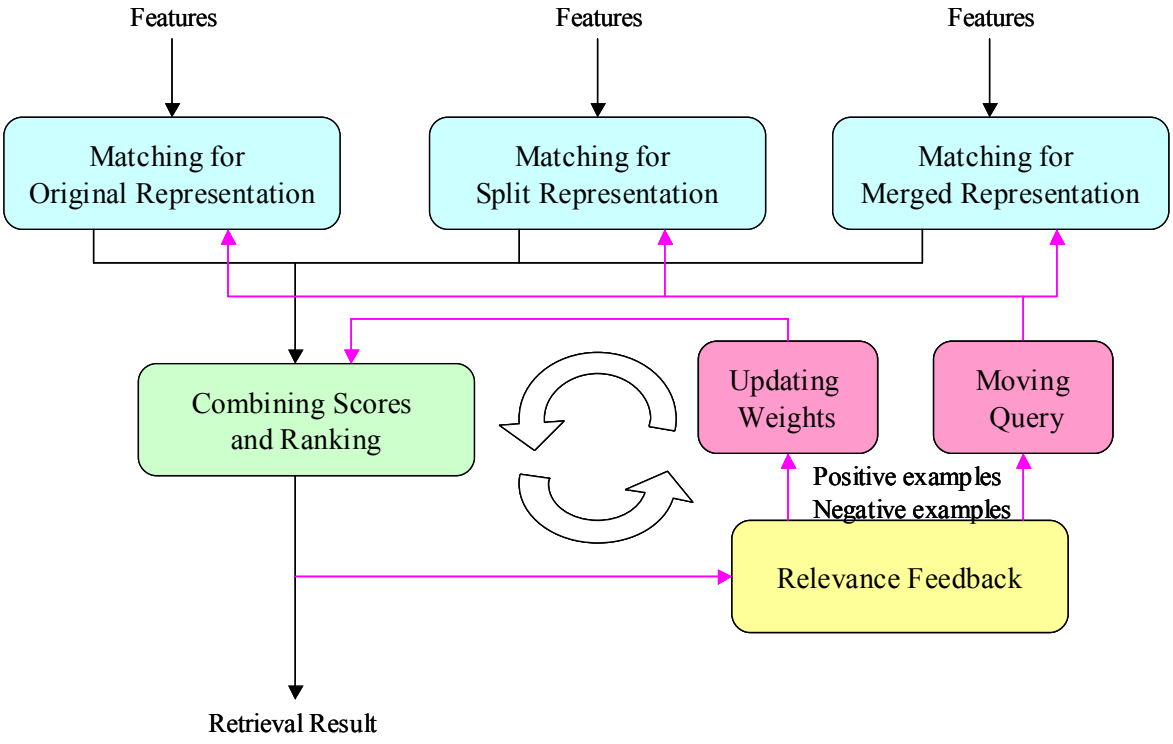


Figure 51 System diagram for relevance feedback

We propose to employ two strategies in the relevance feedback for the sketch retrieval: query feature movement and weight updating. The resulting system diagram for the relevance feedback is shown in Figure 51. The similarity scores from the matching of the multiform representations are combined and the sketches are retrieved based on the ranks of the scores. After the user provides some positive and negative examples, the system updates the features of the query in order to make them more similar to the features of the positive examples and dissimilar to the features of the negative examples. In addition, the system determines which of the multiform representations provides high ranks to the positive examples and low ranks to the negative examples and then increases the weight for the corresponding representation in combining the similarity scores.

This chapter is organized as follows. Section 6.1 starts with the strategy in query feature movement. It first introduces the traditional approach for performing relevance feedback using query feature movement for objects with single component. Next we provide an extension to handle query feature movement for objects with multiple components. Section 6.2 discusses the weight updating approach for relevance feedback. Section 6.3 presents the experimental results using the proposed relevance feedback strategies.

6.1. Query Feature Movement

6.1.1. Object with Single Component

If each object consists of a single component, then in the feature space, each object can be represented by one feature point. One way of performing relevance feedback is to modify the feature point of the query to be closer to the feature points of

the positive examples and farther away from the feature points of the negative examples [48].

Assume
$F_Q^{(i)}$ be the *i*-th feature of the query,
$F_{P_j}^{(i)}$ be the *i*-th feature of the *j*-th positive example,
$F_{N_k}^{(i)}$ be the *i*-th feature of the *k*-th negative example,
where $i = 1,2,\ldots,D$; $j = 1,2,\ldots,n_P$; $k = 1,2,\ldots, n_N$
D is the feature dimension
n_P is the number of positive examples
n_N is the number of negative examples

The features of the query can be modified by the following equation [35]:

$$F_{Q(new)}^{(i)} = F_{Q(old)}^{(i)} + \alpha \sum_{j=1}^{n_P} F_{P_j}^{(i)} - \beta \sum_{k=1}^{n_N} F_{N_k}^{(i)} \quad \text{for } i = 1,2,\ldots,D \tag{36}$$

where α and β are some scaling factors

It can be observed that the final modification of the feature point of the query is obtained by combining the contribution of the feature point from each of the positive and negative examples. Figure 52 illustrates the relevance feedback for objects with a single component.

Chang and Li proposed an algorithm called Maximizing Expected Generalization Algorithm (MEGA) [7] and Tong and Chang proposed to use Support Vector Machine Active Learning [44] to learn the concepts from the relevance feedback for image retrieval. Each object in the database processes a set of concepts that take binary values and the algorithms try to learn which concepts better describe the query such that they will be used as criteria for searching the database. In the sketch retrieval application, it is not intuitive to come up with meaningful concepts to describe a stroke.

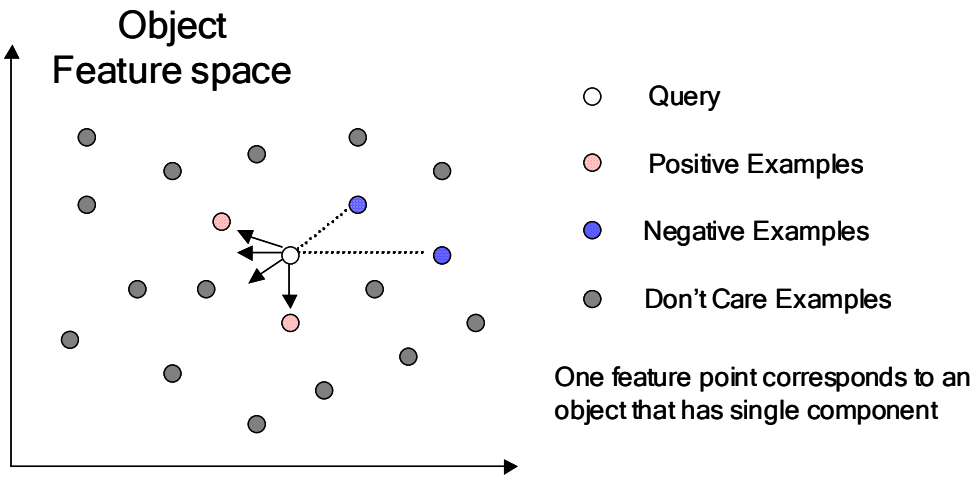


Figure 52 Relevance feedback for objects with a single component

6.1.2. Extension to Object with Multiple Components

The problem will get more complicated when an object may have multiple components. From the relevance feedback, the user will only give positive or negative examples of the objects, and then the system needs to figure out what features from which components are responsible for characterizing the objects that the user wants. We now describe our approach for extending the relevance feedback from objects with one component to objects with multiple components in the next section.

If each object consists of multiple components, then in the feature space, each object can be represented by a variable number of feature points equal to the number of components of that object. Again, relevance feedback can be performed by modifying the feature points of the query to be closer to the feature points of the positive examples and farther away from the feature points of the negative examples. The final modification of the feature point of the query is also be obtained by combining the contribution of the feature point from each of the positive and negative examples. However, there is an additional problem that needs to be solved: which feature point

(component) of a positive or negative example should contribute to the modification of a feature point of the query? In the previous section, it is assumed that the features of the query and the features of the positive and negative examples all have the same dimension D. However, in our system, a query is a sketch that can contain multiple strokes and so are the positive and negative examples. The number of strokes for the query sketch and the number of strokes for each positive and negative example sketch may not be the same. This means that the features of the query and the features of the positive and negative examples can have different dimensions. As a result, equation (36) needs to be modified to account for this situation. Let D_Q, D_{P_j}, D_{N_k} denote the number of feature points in the query, the number of feature points in the j-th positive example and the number of feature points in the k-th negative example respectively. In this case, when we compute the similarity between two sketches that have different number of strokes, the stroke correspondence is first determined and the features of the corresponding strokes are compared. Similarly, by using this stroke correspondence, the features of the query can be modified by considering only the contribution from corresponding strokes if they exist.

$$F_{Q(new)}^{(i)} = F_{Q(old)}^{(i)} + \alpha \sum_{j=1}^{n_P} \sum_{p=1}^{D_{P_j}} M\left(F_Q^{(i)}, F_{P_j}^{(p)}\right) F_{P_j}^{(p)} - \beta \sum_{k=1}^{n_N} \sum_{q=1}^{D_{N_k}} M\left(F_Q^{(i)}, F_{N_k}^{(q)}\right) F_{N_k}^{(q)}$$

$$\text{for } i = 1,2,\ldots,D_Q \tag{37}$$

With this extension, the query feature movement for multiple components can be described by equation (37). Compared with equation (36), it can be observed that there are additional terms $M\left(F_Q^{(i)}, F_{P_j}^{(p)}\right)$ and $M\left(F_Q^{(i)}, F_{N_k}^{(q)}\right)$ that specify the feature point correspondence. For example, if the i-th feature point of the query matches with the p-th

feature point of j-th positive example, then $M\left(F_Q^{(i)}, F_{P_j}^{(p)}\right)=1$. It should also be noted that for each feature point of the query, a maximum of one match is allowed from each example. As a result, there is an underlying constraint given by (38):

$$\sum_{p=1}^{D_{P_j}} M\left(F_Q^{(i)}, F_{P_j}^{(p)}\right) \le 1, \qquad \sum_{q=1}^{D_{N_k}} M\left(F_Q^{(i)}, F_{N_k}^{(q)}\right) \le 1, \qquad i = 1,2,\ldots,D_Q \tag{38}$$

Figure 53 illustrates the relevance feedback for objects with multiple components.

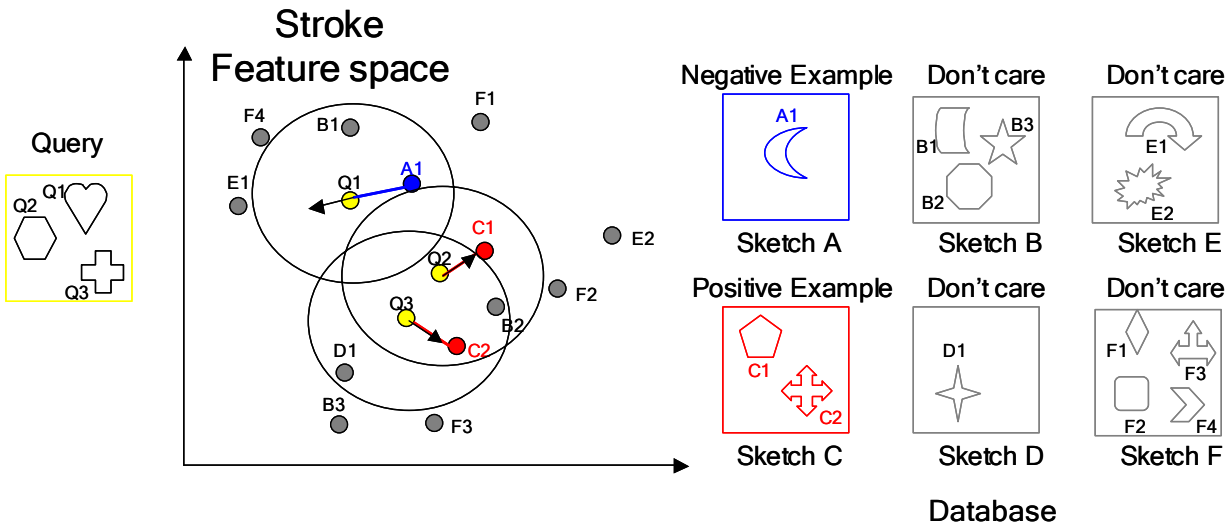


Figure 53 Relevance feedback for objects with multiple components

6.2. Weight Updating

The similarity scores of the multiform representations between two sketches are combined linearly. At first the weights are initialized with equal values and we would like to use the user feedback in order to adjust the weights such that we will increase the weight corresponding to the representation that provides a better retrieval performance. Let P_i denotes the i-th positive examples and N_i denotes the i-th negative examples from the user feedback. n_P is the total number of positive examples and n_N is the total number

of negative examples. Let j be the index for the *j*-th representation: j = 1 denotes the original representation; j = 2 denotes the split representation; j = 3 denotes the merged representation. The quantity R_j given in (39) is directly proportional to the sum of ranks of the negative examples and inversely proportional to the sum of ranks of the positive examples. Essentially R_j is a measure of retrieval performance for the user feedback examples since when R_j is large, the negative examples have low ranking (rank has large value) and the positive examples have high ranking (rank has small value), showing that this representation leads to good performance. On the other hand, when R_j is small, the negative examples have high ranking (rank has small value) and the positive examples have low ranking (rank has large value), showing that this representation does not lead to good performance. The **max** operator is present to take care of the case when no example has been provided (for example, when there is no negative example, i.e., n_N=0, the numerator will be set to be equal to 1 instead of 0). The weights W_j are updated by normalizing R_j as given in (40).

$$R_j = \frac{\max\left(\sum_{i=1}^{n_N} rank_j(N_i), 1\right)}{\max\left(\sum_{i=1}^{n_P} rank_j(P_i), 1\right)} \qquad j = 1,2,3 \tag{39}$$

$$W_j = \frac{R_j}{\sum_{i=1}^{3} R_j} \qquad j = 1,2,3 \tag{40}$$

6.3. Experiments and Results

Figure 54, Figure 55 and Figure 56 show the retrieval performance of the proposed relevance feedback algorithm with weight updating only, with query moving

only and with both query moving and weight updating respectivly by varying the number of iterations of user feedback. Iteration 0 is the initial retrieval performance when no feedback is applied. It can be seen that as the number of iteration increases, the retrieval performance is also increased. The result converges after the 2 iterations. It can also be seen that the gain with the query moving as shown in Figure 55 is more than the gain with weight updating as shown in Figure 54. By combining both methods, the performance is further improved as shown in Figure 56.

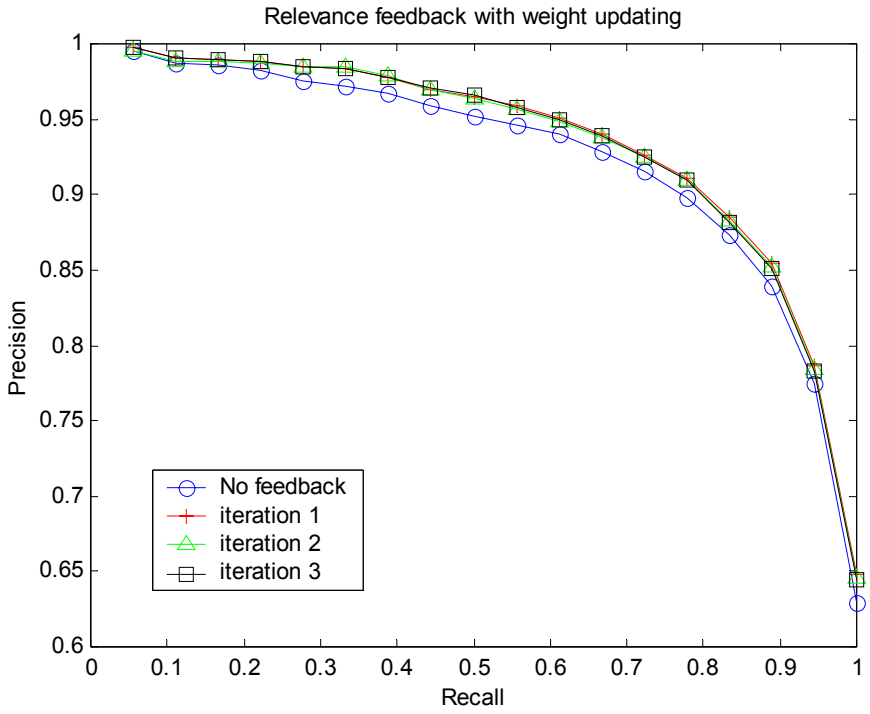


Figure 54 Retrieval performance for relevance feedback with only weight updating by varying number of iterations

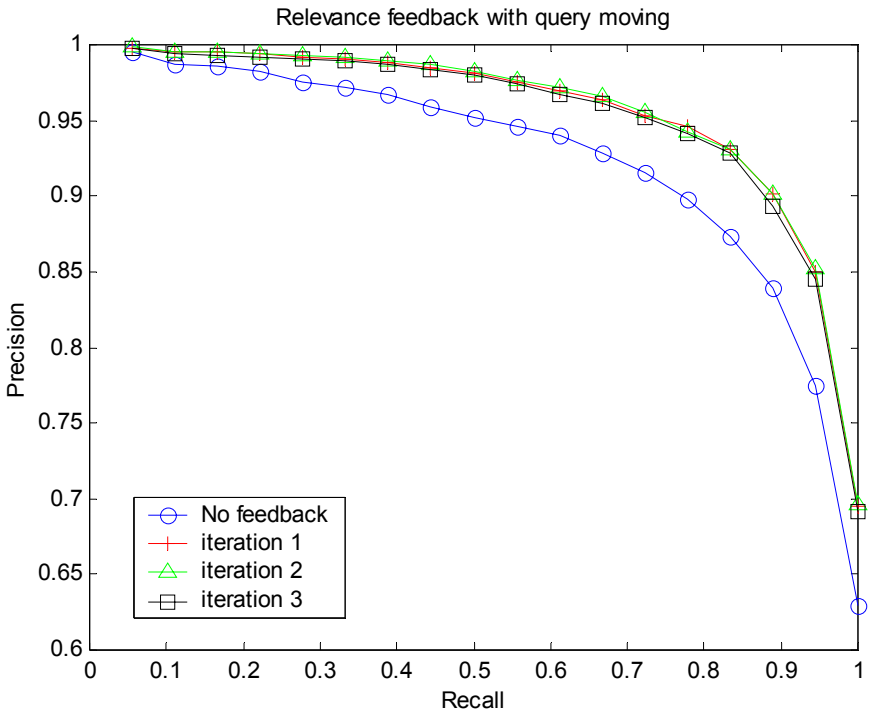


Figure 55 Retrieval performance for relevance feedback with only query moving by varying number of iterations

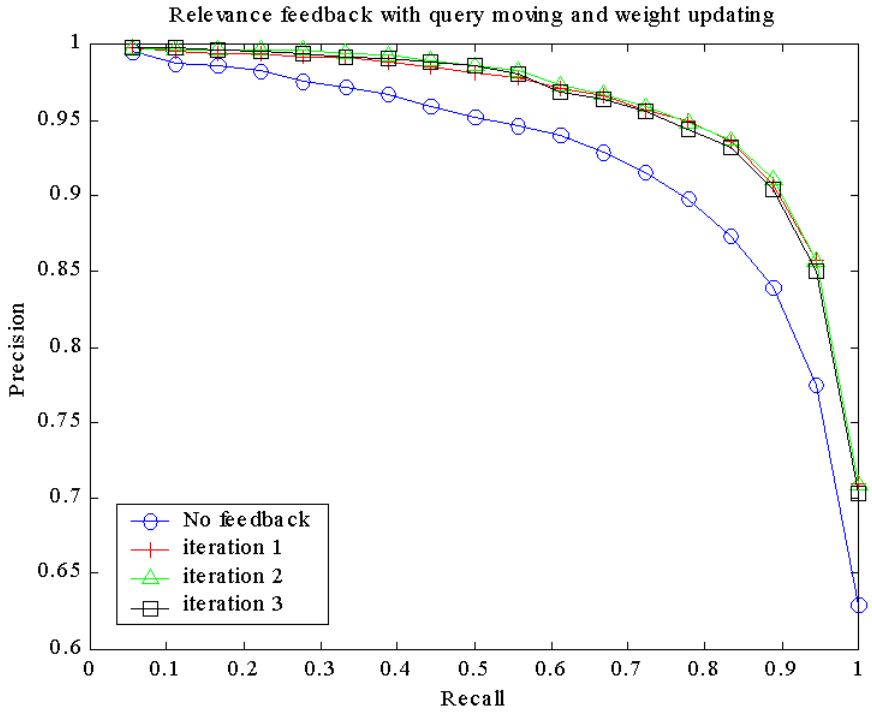


Figure 56 Retrieval performance for relevance feedback with both query moving and weight updating by varying number of iterations

Figure 57, Figure 58 and Figure 59 show the retrieval performance of the proposed relevance feedback algorithm with weight updating only, with query moving only and with both query moving and weight updating respectively by varying the number of feedback examples after 1 iteration. The feedback examples are taken as the top retrieved results at iteration 0 therefore they may include both positive or negative examples. It can be seen that the retrieval performance is increased if more examples are used for the relevance feedback. It can also be seen that the gain with the query moving as shown in Figure 58 is more than the gain with weight updating as shown in Figure 57. By combining both methods, the performance is further improved as shown in Figure 59.

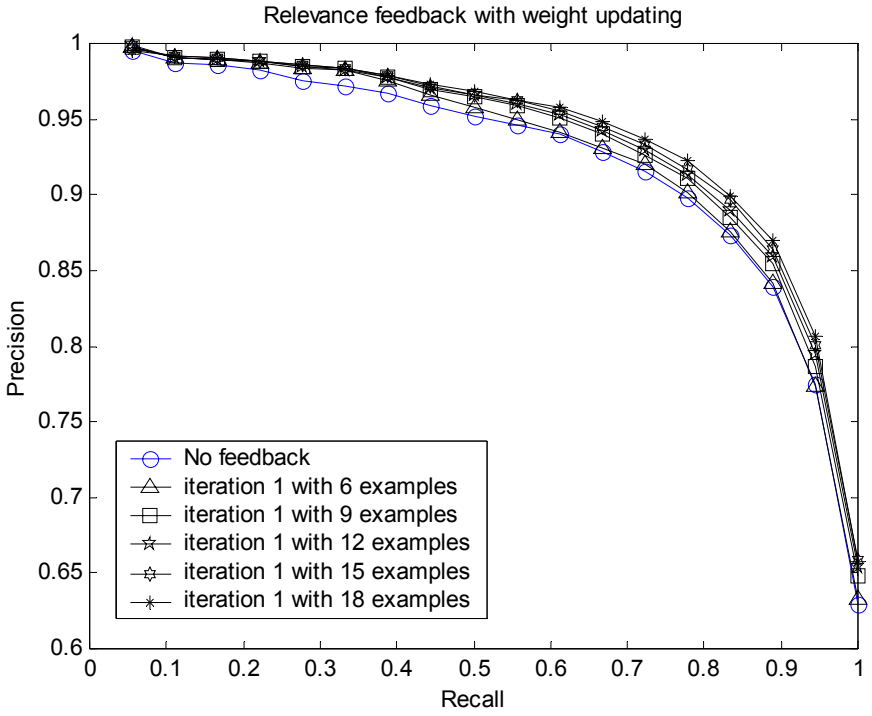


Figure 57 Retrieval performance for relevance feedback with only weight updating by varying number of examples

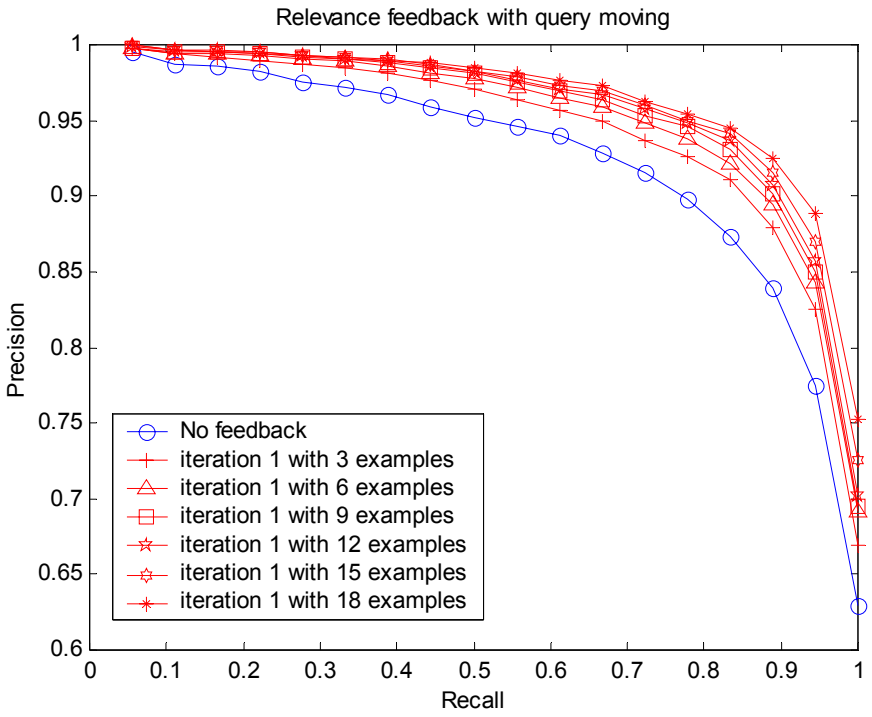


Figure 58 Retrieval performance for relevance feedback with only query moving by varying number of examples

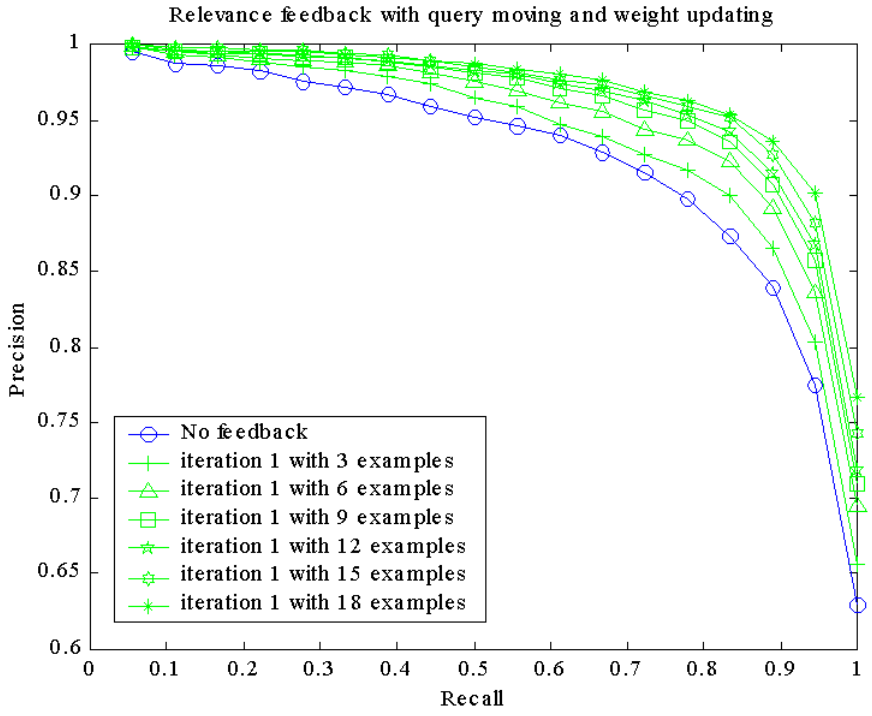


Figure 59 Retrieval performance for relevance feedback with both query moving and weight updating by varying number of examples

7. Partial Matching

This chapter describes our work in *partial* matching that has a different problem setup from *whole* matching. Whole matching is suitable for applications such as trademark retrieval with query by sketch [24], i.e., when the query sketch and the relevant sketch in the database have roughly the same number of strokes as shown in Figure 60. The algorithm is also robust to distortion when there are missing strokes. However, in the case of partial matching as shown in Figure 61, the query sketch may be only a portion of the sketch in the database. In this case, there are much more strokes in a sketch from the database than the query sketch so it is more likely to find a set of strokes that are matched based on the shape. This may lead to a high similarity score even for irrelevant sketches, thus reducing the retrieval precision.

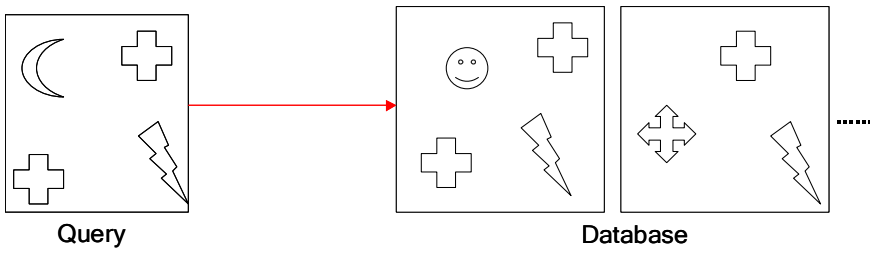


Figure 60 Retrieval with whole matching

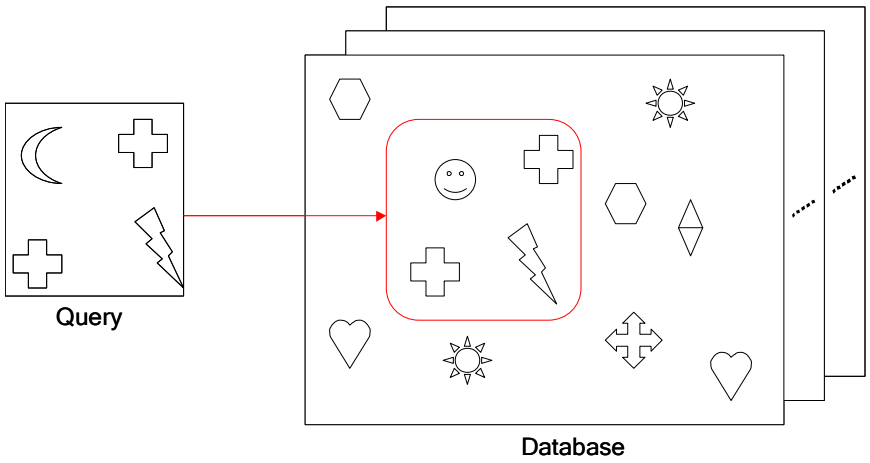


Figure 61 Retrieval with partial matching

As a result, we consider the spatial relation together with shape features when we try to find the stroke correspondence. Specifically, we discuss two approaches that we have considered along this direction. The first approach transforms the stroke correspondence problem into the string matching problem and then uses dynamic programming to solve it. The second approach uses the nearest-neighbor method to find the correspondence between bistroke feature points that consist of shape features of two strokes and the spatial relation between them.

Sketch retrieval with partial matching is useful to find relevant information after jotting and storing notes with a pen-based device. For example, in a classroom, the teacher may write and draw the lecture notes on the whiteboard that can be captured and stored page by page. Later students can retrieve relevant pages from the lecture sketch database by drawing a simple query. For example, a student can draw the chemical structure of benzene as the query and then the system will retrieve the page that contains a similar chemical structure with the associated description about its name, chemical formula and properties. With the partial retrieval capability, it is not necessary to perform segmentation of a page into sketches before the matching, i.e., we do not need to

know which regions of the page form a unit and then decide which regions to be matched with the query.

This chapter is organized as follows. Section 7.1 provides the explanation of two matching schemes we have considered in determining the correspondence of the features. The first approach is dynamic programming and the second approach is bistroke feature matching. Section 7.2 presents the experimental results for partial matching.

7.1. Matching Schemes

7.1.1. Dynamic Programming

By projecting the strokes in the horizontal and in the vertical direction, the strokes can be ordered to form a 1D sequence in each direction. If we denote each stroke by an alphabet, then essentially the stroke correspondence problem can be transformed into the string matching problem. Specifically, we represent each stroke by two alphabets in each direction, and then sort the alphabets according to the x or y coordinates of its boundaries. For a sketch, in each direction, we obtain a string that is a sequence of alphabets sorted according to the spatial locations of the boundaries of the underlying strokes. Now we need to determine the matching (stroke correspondence) between two strings in each direction. String matching can be solved using the dynamic programming technique. First the similarity score table is constructed between every element of the first string and every element of the second string based on the shape similarity between the two underlying strokes. Then starting from the bottom right corner, each element in the similarity score table is updated according to the path that returns a higher gain in score. After constructing updating the entire table, we search for the element in the table with the maximum score and then backtrack to find the path ending in this element. The

resulting path specifies the stroke correspondence and the maximum score in the table is used as the similarity score between two sketches.

7.1.2. Bistroke Matching

A bistroke feature consists of stroke features of a pair of strokes together with the spatial relation between them. As a result, the query with N strokes will correspond to a set of $\binom{N}{2}$ points in the bistroke feature space. The goal of the bistroke feature matching is to find a correspondence between the set of points of the query and the set of points of the page in the bistroke feature space. We can use the following algorithm to find the one-to-one correspondence between the bi-stroke features. Assume that there are N strokes in the query sketch and M strokes in the page, the similarity matrix will contain $\binom{N}{2}\times\binom{M}{2}$ elements. We search for the element in the similarity matrix with the highest score and then its row index and its column index will indicate the corresponding stroke feature points between the query and the page. The similarity matrix is updated by removing the row and the column containing that element. This process is repeated until the similarity matrix is empty. However, the complexity of the bistroke feature matching will become $O\left(\binom{N}{2}\binom{M}{2}\left(\log\binom{N}{2}+\log\binom{M}{2}\right)\right)=O\left(N^2M^2\left(\log N+\log M\right)\right)$ which is too computation intensive. As a result, instead of finding a one-to-one correspondence between the sets of bistroke feature points, we will simply find the nearest bistroke feature point in the page for each of the bistroke feature point in the query. The complexity of the bistroke feature matching is thus reduced to $O\left(\binom{N}{2}\binom{M}{2}\right)=O\left(N^2M^2\right)$.

Although it is now possible for multiple bistroke feature points from the query to correspond to a single bistroke feature point from the page, it is unlikely for this to happen too often since there are relatively large number of bistroke feature points located in a high dimensional space. As a result, the requirement of the one-to-one correspondence for bistroke feature matching may not be as strict as the case for stroke feature matching.

7.2. Experiment and Results

For the experiment, we use those sketches that consist of at least two strokes as the queries. The objective of this experiment is to evaluate the retrieval performance of our proposed algorithm for partial matching. Each element in the database is constructed by combining sketches from different classes to form a page. In each page, we randomly select seven sketches belonging to different classes from the initial collection and put them together by translation and scaling to form a page. It should be noted that the query sketches and the pages of sketches in the database are collected at a different time in order to simulate an actual retrieval scenario in which the database is collected first and then retrieval is performed at a later time. In this database, there are 100 pages in total. For each class of sketch, there are 20 pages in the database that contain a sketch from that class. Figure 62 shows two example pages in the database that contain a sketch from the class "fish". For each query, we retrieve the elements from the database in the descending order of similarity scores.

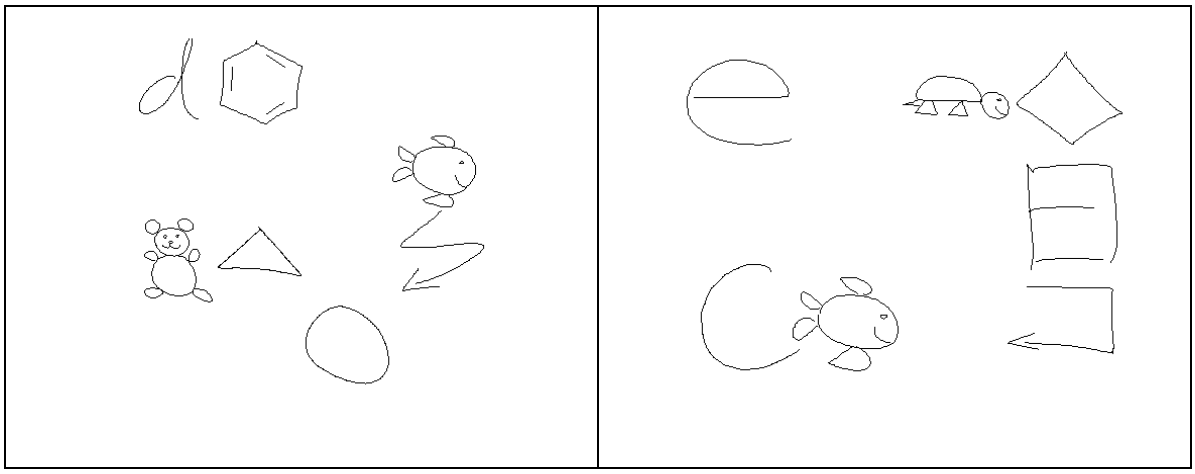


Figure 62 Two example pages in the database

We compare our retrieval result with several other approaches. From Figure 63, it can be seen that the retrieval performance using the dynamic programming approach is very low. With the seven Hu moment invariants [15] as features, the retrieval performance is also very low because these features are more suitable for global matching. By matching the histogram of edge directions, the result is better than the previous approaches. Stroke feature matching only uses the shape features without the spatial relations to find the correspondence. With this approach, there is a significant improvement in the retrieval performance. By using the bistroke feature matching, the retrieval performance is further improved compared with the stroke feature matching. Quantitatively, the stroke feature matching shows an improvement of 62% increase in terms of the average precision compared with the edge histogram matching while the bistroke feature matching shows an additional gain of 56% over the stroke feature matching.

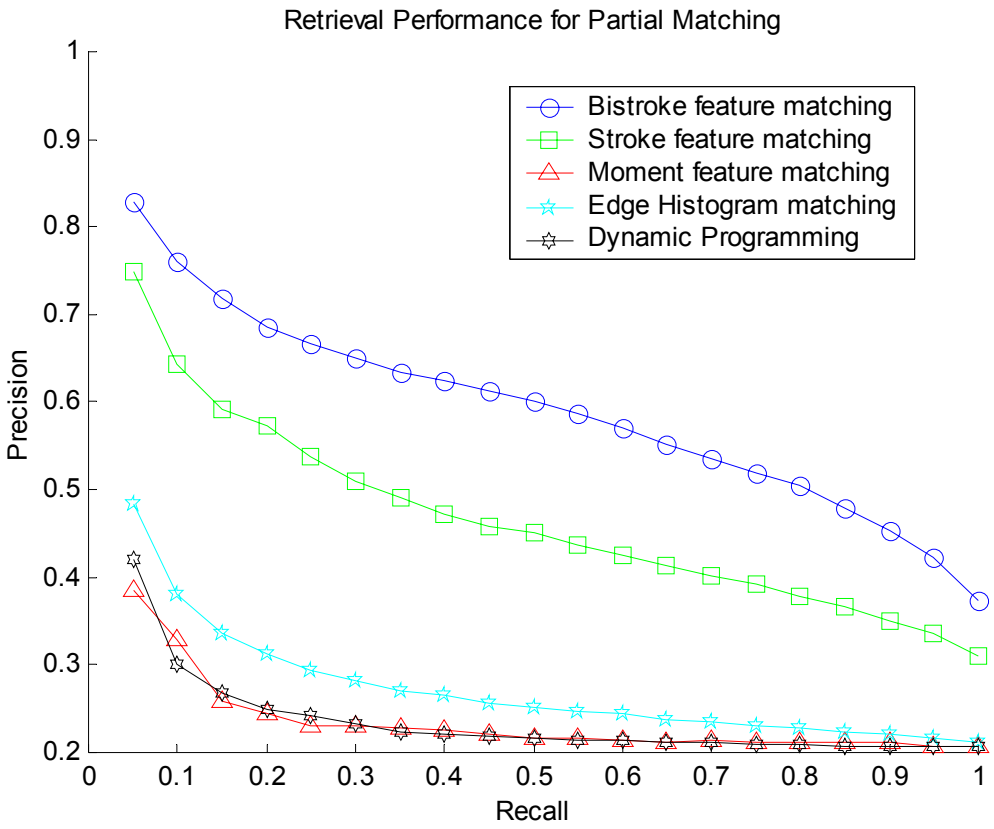


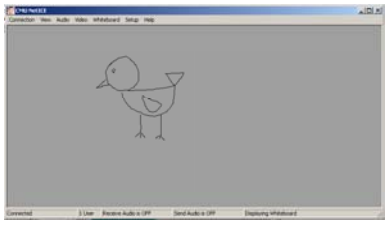
Figure 63 Retrieval performance for partial matching

8. Applications

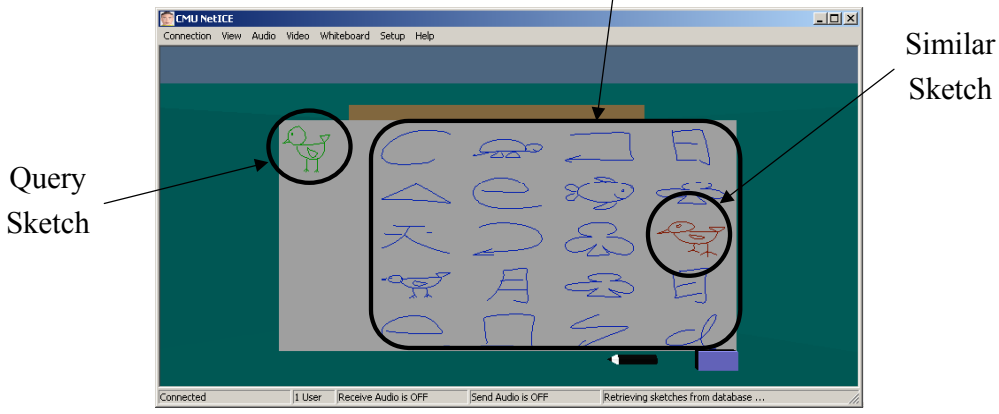
This chapter provides two example applications that perform retrieval based on sketches. Section 8.1 introduces our prototype in sketch retrieval on virtual whiteboard. Section 8.2 describes our prototype in trademark retrieval.

8.1. Sketch Retrieval for Virtual Whiteboard

We have implemented a prototype for free-form hand-drawn sketch retrieval as shown in Figure 64. The user can sketch a query on the whiteboard, and then the system will retrieve those sketches from the database that look similar to the query sketch.



User sketches a query



User retrieves similar sketches from the database

Figure 64 Prototype of free-form hand-drawn sketch retrieval system

8.2. Trademark Retrieval

Our trademark retrieval interface is shown in Figure 65. After the user selects a database a trademarks, he/she can browse through the trademarks. A window will pop up after the user right clicks on the trademark image and the corresponding sketch extracted for that trademark will be shown. Moreover, he/she can click on the trademark to select that trademark to be the query and then the trademarks in the database that are similar to the query will be retrieved. The query is shown on the frame in the left hand side. The twelve trademarks that have the highest ranks according to the similarity score are shown on the frame in the right hand side. In addition to trademark image, the user is also able to provide a sketch as a query, and the trademarks whose corresponding extracted sketches similar to the query sketch are retrieved and ranked as shown in Figure 65. Note that the query is merely a rough sketch yet the system is able to retrieve all the relevant trademarks with the highest similarity scores.

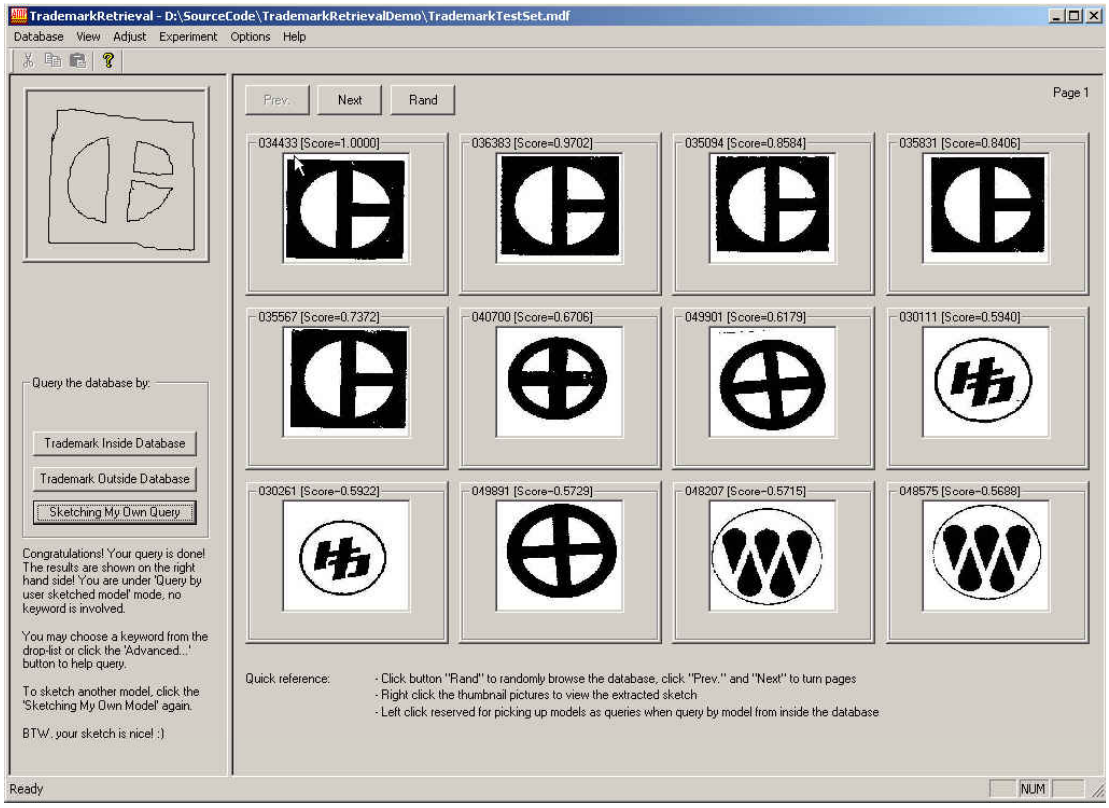


Figure 65 Trademark retrieval user interface

9. Summary and Future Directions

All the work in this thesis is finding an efficient sketch retrieval method to improve the retrieval performance in terms of archiving relevant materials from the database with minimum number of trials. We propose novel approaches along different stages of the retrieval system. Specifically, we target on creating multiform representations of a sketch in order to find at least one consistent representation when it is compared against other similar sketches under user variations. We also target on coarse-to-fine feature extraction in order to capture the characteristics of the sketch at various levels. Our next target is on global and local matching in order to compare various levels of features. Our final target is on multiple component relevance feedback in order to refine the retrieval result based on the user feedback.

The first contribution of this thesis is in preprocessing the sketches for creating multiform representations in order to have certain degree of consistency under each representation. From the original representation of a sketch, we split the strokes into smaller stroke segments based on the dominant points to obtain the split representation. Then from the split representation, we merge the stroke segments if they form a primitive shape to become the merged representation. Since there can be so many possibilities in terms of variations, finding a unique representation that is consistent over all kinds of variations is a difficult task, sometimes there may not be a solution. Therefore, instead of keeping only one representation, we propose to use multiform representations for each sketch such that it is more robust under different kinds of variations. We have shown that

by keeping the multiform representations, the sketch retrieval performance is higher than any of the representation used alone.

The second contribution of this thesis is in extracting high-level features for associating them with semantic meaning. We propose to detect shaded regions and represent them in a coarse level by treating them as one unit. We have built a classifier in this shade detection problem for both hand-drawn sketches and for images. We also propose to construct a stroke hierarchy to keep the structural information of the sketch. By making use of the stroke hierarchy, we introduce a novel concept of hyper-stroke that consists of a group of strokes inside a region enclosed by a boundary stroke. Finer features such as shape features of basic strokes and spatial relations between strokes are also extracted. In addition, we propose to analyze the strokes into primitive shapes. This coarse-to-fine feature extraction can be conceptually considered as looking at the sketch at different points of view. When coarse features are extracted, it can be considered as the case when the sketch is being viewed far away and when fine features are extracted, it can be considered as the case when the sketch is being viewed nearby. We realize that comparing features at different levels allows matching to be performed both globally and locally. While a sketch consists of multiple components and two sketches may have different number of components, we transform the multiple component correspondence problem into the bipartite graph matching problem. We analyze two ways in how this correspondence problem can be solved. We have shown which similarity functions to use for different levels of features and how to combine them to give the overall similarity score.

Another contribution of this thesis is in extending traditional relevance feedback approach from objects with single component to objects with multiple components. We have proposed to use the correspondence in order to move the query feature to make them closer to the positive examples and farther away from the negative examples provided by the user. We have also proposed to update the weights for the similarity scores from the multiform representations according to how much impact they have on the feedback examples. We have shown that the relevance feedback converges at the 2nd iteration and the retrieval performance increases with more examples feedback to the system.

We have considered two matching schemes, dynamic programming and bistroke correspondence, in order to find the correspondence between features for partial matching applications. We have shown that bistroke matching results in 56% gain in the retrieval performance in terms of the average precision compared with stroke matching where spatial relation is not considered in determining the correspondence.

Several prototypes of sketch retrieval system have been implemented targeting on different applications. One prototype system is for the retrieval of hand-drawn sketches on a virtual whiteboard. The scenario is that the teacher first draws the lecture notes on the virtual whiteboard and later a student can retrieve relevant materials by drawing a sketch query so that this application is suitable for distance learning. Another prototype system is for the trademark retrieval. The scenario is that when a person wants to register a new trademark, the clerk in the patent and trademark office can draw a query sketch of the design and use it to search from the trademark database to verify whether similar trademark has already been registered.

There are several possible extensions for the work described in this thesis. It would be interesting to study how to extract more semantic information from a sketch so that we will be able to associate sketches with abstract ideas and retrieve them. In terms of multiform representations, currently we have three representations for a sketch. It may be extended to allow more representations in order to further increase the robustness during matching. Eventually if the computation becomes so fast that perhaps a continuous representation can be built for each sketch and a consistent representation between two sketches can be searched in real time during matching. In our system, the feature extraction mainly focuses on the geometric features. It may be extended to include color features so that it may be more beneficial when trying to retrieve images. Currently the query feature movement strategy in the relevance feedback only operates on the shape features. It can be extended to update other features such as the stroke hierarchy and the hyper-stroke features. It would be interesting to examine how this approach can be further extended to update spatial relations.

Bibliography

[1] Acer Inc., Tablet PC, *http://aac.acer.com/*.

[2] Ansari N. and Delp E. J. "On detecting dominant points". *Pattern Recognition*, Vol. 24, no. 5, pp. 441-451, 1991.

[3] Apte A., Vo V. and Kimura T. D. "Recognizing Multistroke Geometric Shapes: An Experimental Evaluation". In *ACM Symposium on User Interface Software and Technology*, pp. 121-128, 1993.

[4] Aref W., Barbara D., Lopresti D. and Tomkins A. "Ink as a first-class datatype in multimedia databases". In S. Jajodia and V. S. Subrahmanian, editors, *Multimedia Databases*. Springer-Verlag, New York, 1995.

[5] Burkard R.E. and Cela E.. "Linear Assignment Problems and Extensions". In P.M. Pardalos and D.-Z. Du, editors, *Handbook of Combinatorial Optimization*, Dordreck: Kluwer Academic Publishers, pp. 75-149, 1999.

[6] Carson C., Thomas M., Belongie S., Hellerstein J.M. and Malik J. "Blobworld: A system for region-based image indexing and retrieval". In *3rd Intl. Conf. On Visual Information Systems*, Amsterdam, Netherlands, pp.509-516, Springer, 1999.

[7] Chang E. and Li B. "MEGA - The Maximizing Expected Generalization Algorithm for Learning Complex Query Concepts". *ACM Transaction on Information Systems* (final revision), 2002.

[8] Chang S. K., Shi Q. Y. and Yan C.W. "Iconic indexing by 2-D Strings". In *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-9, No. 3, pp. 413-428, May 1987.

[9] Chang S.-K. *Principles of Pictorial Information Systems Design*, Prentice Hall Intern. Editions, 1989.

[10] Cheng Y.-Q., Wu V., Collins R.T., Hanson A.R. and Riseman E.M., "Maximum-weight bipartite matching technique and its applications in image feature matching", In *SPIE Conference on Visual Communication and Image Processing*, 1996.

[11] Del Bimbo, A and Pala, P. "Visual image retrieval by elastic matching of user sketches". In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 2, pp. 121-132, February 1997.

[12] Egenhofer M. J. and Al-Taha K. K. "Reasoning About Gradual Changes of Topological Relationships". In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Frank A. U., Campari I. And Formentini U., eds., Lecture Notes in Computer Science no. 693, pp. 196-219, Springer-Verlag, 1992.

[13] Electronics Imaging Inc., E-Beam, http://www.e-beam.com/.

[14] Faloutsos C., Barber R., Flickner M., Niblack W., Petkovic D. and Equitz W. "Efficient and Effective Querying by Image Content". In *Journal of Intelligent Information Systems*, 3:231-262, 1994.

[15] Hu M. "Visual Pattern Recognition by Moment Invariants". *IRE Trans. on Information Theory*, vol. IT-8, 1962.

[16] Jacobs, C. E., Finkelstein A., and Salesin D. H. “Fast Multiresolution Image Querying”. *Proceedings of SIGGRAPH '95*, pp. 277-286, 1995.

[17] Jing F., Li M., Zhang L., Zhang H.-J. and Zhang B. “Learning in Region-Based Image Retrieval”. *Proc. Intl. Conf. on Image and Video Retrieval*, 2003.

[18] Kamel I. and Barbara D. “Retrieving Electronic Ink by Content”. In *Proc. of the Intl. Workshop on Multimedia Database Management Systems,* pp. 54-61, 1996.

[19] Kato T., Kurita T., Otsu N. and Hirata K. “A Sketch Retrieval Method for Full Color Image Database”. In *Proc. of the 11th Intl. Conf. On Pattern Recognition*, pp. 530-533. The Hgues, The Neitherlands, August 1992.

[20] Korn F., Sidiropoulos N., Faloutsos C., Siegel E. and Protopapas Z., “Fast and Effective Similarity Search in Medical Tumor Databases using Morphology”. *SPIE Proceedings* Vol. 2916, Boston MA, Nov. 1996.

[21] Kuhn, H. W., “The Hungarian Method for the Assignment Problem”, Naval Research Logistics Quarterly, Vol. 2, 1955, pp. 83-97.

[22] Lam L., Lee S.-W. and Suen, C.Y.. “Thinning methodologies – a comprehensive survey”. *IEEE Tran. on Pattern Analysis and Machine Intelligence*, Vol. 14: 9, pp. 869-885, September 1992.

[23] Leung W. H. and Chen T. “Retrieval of Sketches Based on Spatial Relations Between Strokes”. *IEEE Intl. Conf. on Image Processing*, Vol. 1, pp. 908-911, Rochester, NY, September 2002.

[24] Leung W. H. and Chen T. “Trademark Retrieval with Contour vs Skeleton Classification”. *IEEE Intl. Conf. on Multimedia and Expo.*, Lausanne, Switzerland, August 2002.

[25] Leung W. H. and Chen T. "User-Independent Retrieval of Free-Form Hand-Drawn Sketches". *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 2029-2032, Orlando, FL, May 2002.

[26] Liu C.-L., Kim I.-J. and Kim J H. "Model-based stroke extraction and matching for handwritten Chinese character recognition". *Pattern Recognition,* 34, pp. 2339-2352, 2001.

[27] Loncaric S. "A survey of shape analysis techniques". In *Pattern Recognition*, 31(8), pp. 983-1001, 1998.

[28] Long A. C., Landay, J. A., Rowe, L. A. and Michiels J. "Visual similarity of pen gestures". In *CHI 2000, ACM Conf. on Human factors in Computing Systems*, 2(1), pp. 360-367, April 2000.

[29] Lopresti D. and Tomkins A. "Approximate Matching of Hand-Drawn Pictograms". In *Proc. Third Int. Work. Frontiers Handwriting Recognition*, pages 102-111. May 1993.

[30] Lopresti D. and Tomkins A. "On the searchability of electronic ink". In *Proc. of the Fourth Intl. Workshop on Frontiers in Handwriting Recognition*, pp. 156-165, December 1994.

[31] Lopresti D. and Tomkins A. "Temporal domain matching of hand-drawn pictorial queries". In *Proc. of the Seventh Conf. of The Intl. Graphonomics Society*, pp. 98-99. August 1995.

[32] Lopresti D., Tomkins A. and Zhou J. "Algorithms for matching hand-drawn sketches". In *Proc. of the Fifth Intl. Workshop on Frontiers in Handwriting Recognition*, pp. 233-238, September 1996.

[33] Matusiak S., Daoudi M. and Blu T. "Sketch-Based Images Database Retrieval". In *Proc. of the Fourth Intl. Workshop on Multimedia Information Systems (MIS'98)*, pp. 185-191, Istanbul, Turkey, September 1998.

[34] Microsoft, NetMeeting 3, videoconferencing software, http://www.microsoft.com/windows/netmeeting/..

[35] Nabil M., Ngu A. H. H. and Shepherd J. "Picture Similarity Retrieval Using the 2D Projection Interval Representation". In *IEEE Trans. on Knowledge and Data Engineering*, Vol. 8, No. 4, pp. 533-539, August 1996.

[36] Nabil M., Shepherd J. and Ngu A. H. H. "2D Projection Interval Relationships: A Symbolic Representation of Spatial Relationships". In *Advances in Spatial Databases: Fourth Int'l Symp., SSD '95*, Lecture Notes in Computer Science no. 951, pp. 292-309, Springer-Verlag, 1995.

[37] Pentland, A., Picard, R. W., Sclaroff, S., "Photobook: Content-Based Manipulation of Image Databases". *SPIE Storage and Retrieval Image and Video Databases II*, 1994.

[38] Petrakis E. G.M. and Faloutsos C. "Similarity Searching in Medical Image Databases". *IEEE Trans. on Knowl. and Data Eng.*, 9(3):435–447, May/June 1997.

[39] Rocchio, J. J. "Relevance Feedback in Information Retrieval". In *The SMART Retrieval System, Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1971.

[40] Schomaker, L. "From handwriting analysis to pen-computer applications". In *Electronics & Communication Engineering Journal*, Vol. 10: 3, pp. 93-102, June 1998.

[41] Sciascio E. D. and Mongiello M. “Query by Sketch and Relevance Feedback for Content-Based Image Retrieval over the Web”. In *Journal of Visual Languages and Computing*, 10(6), pp. 565-584, 1999.

[42] Seiko Instruments USA Inc., SmartPad, http://www.seikosmart.com/products/sp580.html.

[43] Smith J. R. and Chang S.-F. “Integrated Spatial and Feature Image Query”. *Multimedia Systems*, 7:129–140, 1999.

[44] Tong S. and Chang E. “Support Vector Machine Active Learning for Image Retrieval”. *ACM international Conference on Multimedia*, October 2001.

[45] Tsay Y. and Tsai W.. “Attributed string matching by split and merge for on-line Chinese character recognition”. In *Pattern Recognition and Machine Analysis*, Vol. 7, pp. 453-462, 1985.

[46] Tseng B., Shae, Z.-Y., Leung W. H. and Chen, T., "Immersive Whiteboards in a Networked Collaborative Environment", *IEEE Intl. Conf. on Multimedia and Expo.*, Tokyo, August 2001.

[47] van Rijsbergen C. J. *Information Retrieval,* Butterworths, London, 1979.

[48] Veltkamp R.C. and Hagedoom M. “State of the Art in Shape Matching”. *Technical Report UU-CS-1999-27*, Ultrecht University, the Netherlands, 1999.

[49] Virtual Ink Inc., Mimio, Virtual Ink Corporation, mimio. [Online]. Available: http://www.mimio.com/.

[50] WACOM Technology co., graphire2 tablet, http://www.wacom.com/graphire/.

[51] Wang Y.-H. “Image indexing and similarity retrieval based on A new Spatial Relation Model”. In *Intl. Conf. on Distributed Computing Systems Workshop*, pp. 16-19, April 2001.

[52] Y. Hara, A. M. Keller, G. Wiederhold, “Implementing Hypertext Database Relationships through Aggregations and Exceptions”. *Proc. of 3rd ACM Conference on Hypertext*, pp. 75-90, San Antonio, Texas, Dec 15-18, 1991.