

COSC 460 Project: Shape Based Image Retrieval

November 3, 2000

Michael Brown

Abstract

A new technique is proposed for representing shape features for the purpose of image retrieval. This project defines the properties of this representation, and implements software that extracts the relevant features from a given image and converts them into a recognised format. The project also implemented testing software that enables queries to be performed on a medium sized database of images and evaluates the performance of the proposed representation. The representation achieved 82% *precision* and 83% *recall*.

Contents

1	Introduction	4
1.1	Report Structure	4
2	Background	6
2.1	One-Dimensional Shape Representation	6
2.2	Two-Dimensional Shape Representation	6
2.3	Regional Descriptors	7
2.4	Moments	7
2.5	Fourier Theory	7
2.6	Histogram Based Approaches	7
2.7	Syntactical methods	8
3	The Representation	9
3.1	Properties of a Good Representation	9
3.2	Overview of Our Representation	10
3.3	The Finer Points of the Representation	11
3.3.1	Separation of Curves	11
3.3.2	Complex Objects	12
3.3.3	Curves Greater Than 180 degrees	13
3.3.4	Length Normalisation	14
3.3.5	Starting Point	14
3.3.6	Direction	14
3.4	Motivation For Our Representation	15
4	Shape Matching with the Representation	16
4.1	Edit Distance	16
4.2	Statistical Comparisons	17
5	Implementation	19
5.1	Extracting Software	19
5.1.1	Corner Detection	19
5.1.2	Analysis Of Segments	21
5.1.3	Extracting Further Information	21
5.1.4	Converting To Strings	22

5.2	Query Program	23
5.2.1	Comparing Edit Distances	23
5.2.2	Statistical Comparisons	23
5.3	Parameters and Fine Tuning	24
5.3.1	Corner Detection	24
5.3.2	Analysis of Segments	25
5.3.3	Number of Alphabet Symbols	25
5.3.4	Comparing Edit Distances	25
5.3.5	Statistical Comparisons	26
6	Testing	27
6.1	Statistical Features	27
6.2	Exhaustive Starting Point and Direction Search	27
6.3	Experimental Design	28
6.3.1	The Image Database	28
6.3.2	Precision and Recall	28
6.3.3	Ranking	28
6.4	Results	30
6.4.1	Precision and Recall	30
6.4.2	Ranking	31
6.4.3	Comments	31
7	Conclusion	35
7.1	Future Work	35

Chapter 1

Introduction

While most traditional retrieval systems perform searches using comparisons of text based strings, content based systems extract features from the content of an image to judge its similarity with another. There are three main types of features that are extracted from images: colour, texture and shape. The first two approaches have been explored more thoroughly than shape based approaches. The focus of this project is on shape based image retrieval.

Like any other features based on human perception, the major problem in the use of shape is how to represent shape information, and how to describe the shape of an object. The main purpose of a good representation, with respect to shape-based image retrieval, is to perceive similar shapes to be similar and dissimilar shapes to be dissimilar. This should hold through different invariances such as changes in scale and orientation. A new technique for representing shapes for the purpose of image retrieval is proposed. This project defines this representation and demonstrates how it will be used to compare two given shapes. The proposed representation is a boundary based approach that segments a given shape, based on its corner points, into three different primitives.

A program that extracts the proposed representation from images was implemented with a query program that returned ranked results for a given image. The representation's performance was evaluated in terms of how effective the above goal was met by testing the results from queries performed on a sample image database. The test criteria asked two questions: were the most similar images returned from the image database? and were there any dissimilar images returned? The program's performance was judged against a human's ranking of the test images.

1.1 Report Structure

Chapter two provides some background into the problem area, highlighting other previously applied approaches to shape based image retrieval. Chap-

ter three discusses the details and properties of the proposed representation. This is followed by an explanation of the methods used to compare any two extracted representations of an object's shape. Chapter five discusses the implementation of the software that extracts the proposed representation from a shape's boundary, performs the comparisons between any two representations and performs queries on an image database. Chapter six presents the results and evaluation of tests performed using the proposed representation, while the last chapter concludes and discusses future work.

Acknowledgements

I would like to thank Dr Donald Adjero, my supervisor, for all his help and guidance. Much thanks also to Jane McKenzie who helped with the proof reading of this report.

Chapter 2

Background

A literature survey in the area of image retrieval was undertaken. The three main techniques for image retrieval (colour [1, 2, 3, 4, 5], texture [6, 5] and shape [5, 7, 8, 9, 10, 11, 12, 13, 14]) were investigated as well as some practical image retrieval systems [15, 16, 17] with the focus shifting to shape based techniques.

There are a wide range of approaches for representing shape. These can be grouped into two categories: boundary-oriented methods and region-based methods. Boundary-oriented methods gather information about a shape, focusing on its boundary, whereas region-based techniques use information gained from the entire shape, including the interior regions within the shapes boundary.

2.1 One-Dimensional Shape Representation

There are many different examples of one-dimensional representations of a shape's features [14, 5]. For example one method involves representing the shape linearly as angle and radius difference functions taken from the shape's centre of gravity. The idea here is to transform the two-dimensional boundary into a one-dimensional functional representation. Many other one-dimensional approaches work this way and a number of other characteristics can also be used to describe the boundary of the shape, e.g. the so-called shape signatures [5].

2.2 Two-Dimensional Shape Representation

Two-dimensional techniques represent the shape in its two-dimensional form [5]. These are more difficult to describe than a one-dimensional representation, but contain more information, particularly about spatial relationships.

One popular two-dimensional method is the polygonal representation, in which the shape's boundary is represented as piece-wise polynomials defined

using some two dimensional coordinate system. The result is a polygon which provides an approximation of the original shape.

2.3 Regional Descriptors

Some example regional descriptors include area and perimeter. The area of a region is defined as the number of pixels contained within its boundary. The perimeter of a region is the length of its boundary. These descriptors are useful when the size of objects is important.

2.4 Moments

A shape can also be represented quantitatively using moments [5]. Both one and two-dimensional moments can be used for shape representation. These describe features from the shape, for example, if the boundary is transformed into some one-dimensional representation, a moment might provide information about the spread of the shape about the mean of the parameter of the one-dimensional function.

The problem with one-dimensional moment based features is that they do not retain information about spatial relationships in the image.

2.5 Fourier Theory

Some authors have also applied Fourier Theory to shape representation [10]. The shape is described from its boundary with a sequence of complex coefficients known as Fourier descriptors. These coefficients represent the shape of an object in the frequency domain, where the low frequencies symbolise its general contour, while the high frequencies represent details in the contour.

2.6 Histogram Based Approaches

Another method is the histogram representation where some characteristics of a shape are categorised, and counts of each are stored to represent the shape. This, however, does not represent the sequence of these categorised segments and only supplies statistical information about the shape. Histograms are usually normalised against scale. Rotation of an image will result in shifts in the histogram bins. To ensure rotation invariance, it may be necessary to match across all possible shifts.

2.7 Syntactical methods

Syntactical methods represent a shape as a series of symbols from a predefined alphabet. Syntactical boundary based techniques are the most closely related existing methods to the proposed representation.

One such technique, the chain code [5], is one of the most widely used approaches for representing shape. Here the boundary of the shape is coded using a numbering scheme to represent the changes in the shape. A different number is added to the string representing the shape depending on the direction of the change. Such a technique is a generalisation of the polygonal approximation, which breaks down the shape's boundary into segments of straight lines.

Chapter 3

The Representation

A new method of representing shapes for the purpose of effective image retrieval is proposed. This chapter discusses the properties of a good representation as well as defining our representation, its properties and the implications of these.

3.1 Properties of a Good Representation

A good representation is one that will provide a means to perform effective and efficient retrieval on images. The higher level goal of a good representation is to perceive similar shapes to be similar and dissimilar shapes to be dissimilar. This implies that each similar object's representation must be both similar and unique.

Many approaches to shape based representation provide only a statistical means of comparing two shapes. Such techniques lose a lot of the original information from the shape. The project focuses on similarity matching not exact matching. There is a balance between retaining important information while still being general enough not to regard similar objects as dissimilar.

A good representation should exhibit the following properties [14].

- Invariance, i.e., two shapes that are exactly the same should both produce exactly the same representation.
- Uniqueness, i.e., no two dissimilar shapes will have a similar representation.
- It should be able to handle small changes in an object's shape. If the representation is too detailed it may not be very effective in this matter.
- It must be robust to changes in orientation and scale. Any change in these does not imply that the object is different, even though to a computer their images may be dissimilar. It is not very likely that two similar

objects will always appear in images at the same scale or in the same orientation.

3.2 Overview of Our Representation

The proposed representation uses a boundary approach. It represents the boundary of a given shape syntactically as a sequence of decomposed shape segments. The segments consist of three basic primitives: concave and convex curves and a straight line.

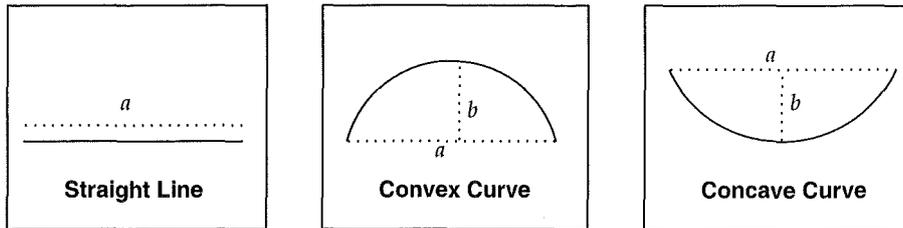


Figure 3.1:

The shape's boundary is divided into segments, each of which are classified as one of these three basic primitives. Additional information is also stored about the dimensions of these segments as well as the angles of the transition from one segment to another. For each of the curves we store the length from their starting point to their end point, (represented as a in Figure 3.1) and the height from the midpoint of this line to the top of the curve (represented as b in Figure 3.1). For a straight line we store its length (represented as a in Figure 3.1). The convex shape bends towards the interior of the shape while a concave curve bends away. The angles (as shown by $\theta_{(1,2)}$ in Figure 3.2) tell us the *internal angle* from the *baseline* of one of the shape's segments to the next. We use a (see figure 3.2) as the *baseline* for the segments.

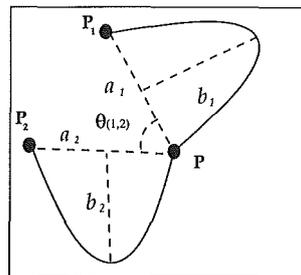


Figure 3.2:

For a given shape, the result will be a decomposition into its basic shape primitives which we can then view as a sequence of symbols. The symbols will be drawn from a special alphabet made up of the three basic shape segments. In effect the shape of any object can then be viewed as a string of symbols [18], and we can then use available string matching methods to compare two shapes.

3.3 The Finer Points of the Representation

There are many problems that must be considered when trying to achieve all the properties of the ideal representation. In implementing the finer points a number of issues must be considered. This section summarises some of the relevant issues that need to be considered in implementing the finer points of our approach and discusses the choices we made.

3.3.1 Separation of Curves

There are a number of ways to segment a shape. One is to segment it at fixed intervals. This is the simplest solution, and is used in similar boundary-based approaches such as the chain code. However, it may not be the most appropriate, as the fixed segments may only cover part of a feature or a couple of different features. The top diagram in Figure 3.3 shows a boundary that is segmented into three fixed intervals.

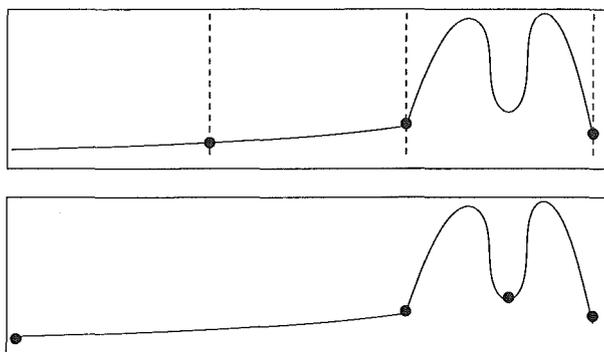


Figure 3.3:

As can be seen, these are not the most logical places to segment the shape. The first two segments mostly cover just one feature, a straight line. The third segment then contains two main features, and would need to be averaged to represent it with the most relevant symbol. This may remove important features from the shape's boundary, particularly in cases where scale is important. The size of the segments also becomes a very important issue with this method.

The second method segments the shape based on its features as in the lower diagram of Figure 3.3. This allows us to select the most relevant points, but is more difficult. The level of detail with this method also becomes a stronger consideration. We work with the latter solution as it provides a closer approximation to the true shape.

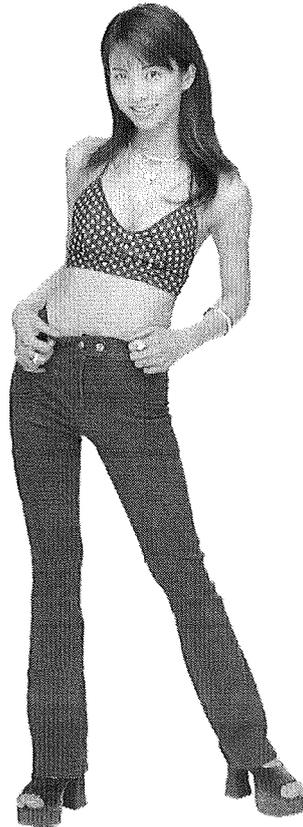


Figure 3.4:

3.3.2 Complex Objects

Determining where an actual object's boundary begins and finishes can be a problem. Consider the case of the person in Figure 3.4. Using edge detection, the boundary of her shape may be found, but within it there may be many more boundaries. For example, the areas where pieces of clothing and skin meet may look like different objects, so instead of representing a person we may be representing a head, top, trousers and some shoes. A similar problem

applies when it is not clear if external areas are part of the object or not, and are falsely included.

We wish to consider how well the representation works on any given object's shape, not how to segment images, therefore, we will use single objects on simple backgrounds so they will be clearly defined.

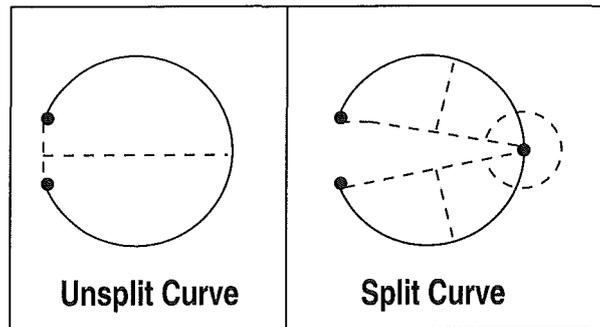


Figure 3.5:

3.3.3 Curves Greater Than 180 degrees

If an object poses a curve that is greater than 180 degrees as in Figure 3.5, it can not be represented by any of the three primitives of our representation. In this case the curve must be split in half. The midpoint of the *baseline* (a) of the curve is taken and a perpendicular line is projected from there to the edge of the curve, where a new point is inserted, resulting in two segments being formed. Figure 3.5 illustrates an example of this technique.

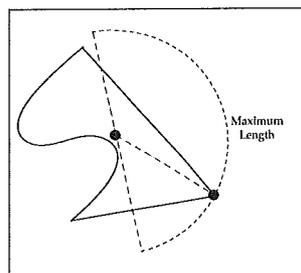


Figure 3.6:

3.3.4 Length Normalisation

To overcome the problem of scale, the representation will store normalised lengths. The length of a segment a will be normalised to l where $l = a \div normfact$ such that $0 < l \leq 1$. The value for $normfact$ is the normalisation factor for which, $normfact \geq l_{max}$, where l_{max} is the longest length of any possible segment of a shape. To ensure this, $normfact$ must be half the circumference of the bounding circle for the shape, i.e. $normfact = \pi \times d_{max}$, where d_{max} is the distance of the furthest point from the centre of gravity. Figure 3.6 illustrates this approach.

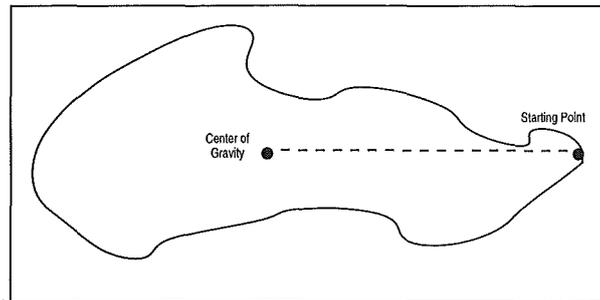


Figure 3.7:

3.3.5 Starting Point

The starting point should always be at the same relative point for any given object. If the representation does not start at the same point each time, it will not form a consistent representation of the object. Our method uses the furthest point from the centre of gravity as the starting point, as shown in Figure 3.7.

This introduces another sub-problem: when there are multiple points that are the same distance from the centre of gravity. To overcome this problem we have devised an algorithm which also helps to determine which direction to travel from the starting point. This algorithm will be discussed in the next section as it also relates to the choice of direction in which the shape's boundary is traversed.

Our representation must not be too strict when selecting the furthest point from the centre of gravity. If two points are a similar distance, they must be considered the same, as two similar shapes may contain different points that are the furthest from the centre of gravity, as shown in Figure 3.7.

3.3.6 Direction

Once a starting point is found, the shape's boundary must be traversed in a consistent manner. If two shapes are identical this will not pose a problem.

However, consider the case when a shape is reflected or altered in some manner. We will need to travel in a different direction for each of the shapes in order to perceive them as similar.

We have developed an algorithm that simultaneously determines both the starting point and the direction in which to traverse the shape from the starting point. The method finds all the potential starting points (those that are the furthest points away from the centre of gravity), then measures the lengths of both the adjoining segments from each of these points. The point with the adjoining segment with the longest length becomes the starting point, and its direction in reference to the point becomes the direction in which the object is traversed. In the case of a tie, the next set of symbols along the same direction of each of the tied elements will then be checked and compared.

Although this algorithm guarantees finding the correct starting point, analysis has shown that this algorithm may not be very effective in all cases when two similar objects vary slightly. An alternative approach is to make an excessive search of all possible starting points and directions each time a representation of one shape is compared with an other. This overcomes the problem but is much more time consuming at the matching phase, which implies longer response time to user queries. We have chosen to use both methods separately to determine the best.

3.4 Motivation For Our Representation

The proposed representation is a boundary based approach similar to the chain code and polygon approximation. However, it offers advantages over both these alternatives.

The chain code uses fixed intervals. The features it extracts do not relate as strongly to the shape's boundary, whereas the proposed representation decomposes segments based on the shape's actual features, therefore perceives the boundary in a natural manner. The chain code can use small intervals, providing a very detailed description. Such detail may describe a single curve with a large number of primitives, whereas the proposed representation could represent it with one. This means that the number of comparisons between primitives to judge similarity will be far fewer.

The polygon approximation only has one primitive, a straight line. Any curves will be represented as multiple lines. This representation does not capture the features as well as a representation that has a primitive to express that feature.

Chapter two discussed many of the different types of shape based representations. The proposed representation was designed with similarity based image retrieval in mind. In doing this the focus was on ensuring the proposed representation had the properties of a good representation as discussed previously. It is this design focus that gives our proposed representation an advantage over some of the previously discussed representations for shape based image retrieval.

Chapter 4

Shape Matching with the Representation

The result of the shape decomposition as described in chapter three will be a string representation of an input shape. Two types of comparisons will be made; firstly comparing the string representations of our shapes, and secondly using statistical data. Both techniques will be used independently for this project, although it would be possible to combine them.

4.1 Edit Distance

An image will be represented by three different strings, each with a different alphabet. The three strings will be for: the type of segments, their lengths, and the adjoining angles between two given segments. The segments will be represented by a three symbol alphabet, one symbol for each primitive. The lengths' and angles' alphabets will be determined by splitting the lengths or angles into intervals of a fixed length to determine their allotted symbol. String pattern matching and ideas from the vString edit distance [18] will be used to compare any two strings.

The edit distance indicates the number of edit operations we need to perform to transform one string into the other. All three strings will be matched. The weighting given to each string for the overall distance also has some importance.

We considered three operations that could be performed on any two strings that would make them differ: *Deletion*, *Insertion* and *Substitution*. More specifically the edit distance between $A : a_1 a_2 \cdots a_n$ and $B : b_1 b_2 \cdots b_m$, $D(A, B)$, is usually determined by the use of recurrence relations:

initialisations:

$$\begin{aligned} D_{0,0} &= 0, \\ D_{i,0} &= D_{i-1,0} + \alpha_{del}(a_i), \end{aligned}$$

$$D_{0,j} = D_{0,j-1} + \alpha - ins(b_j);$$

main recurrence:

$$D_{i,j} = \min \begin{cases} D_{i-1,j} + \alpha_{del}(a_i) & (deletion), \\ D_{i-1,j-1} + \alpha_{subs}(a_i, b_j) & (substitution), \\ D_{i,j-1} + \alpha_{ins}(b_j) & (insertion), \end{cases}$$

where $1 \leq i \leq |A| = n$; $1 \leq j \leq |B| = m$; α_{del} , α_{ins} , and α_{subs} are the respective costs of deletion, insertion, and substitution edit operations.

These three are a basic set of edit operations, but others, such as *swap* and *fusion* have been used for matching [18]. For this project we will only make use of the *deletion*, *substitution* and *insertion* edit operators to match two different sequences for each of the three strings, i.e. due to the the shape segments, lengths and angles, we compute the corresponding edit distance.

4.2 Statistical Comparisons

Statistical information can also be drawn from these strings. Such statistics include the counts of the different segment primitives or the total length of all the segments. Although these measures lose information about the shape, they can still be used to give an approximation if any two given shapes are similar.

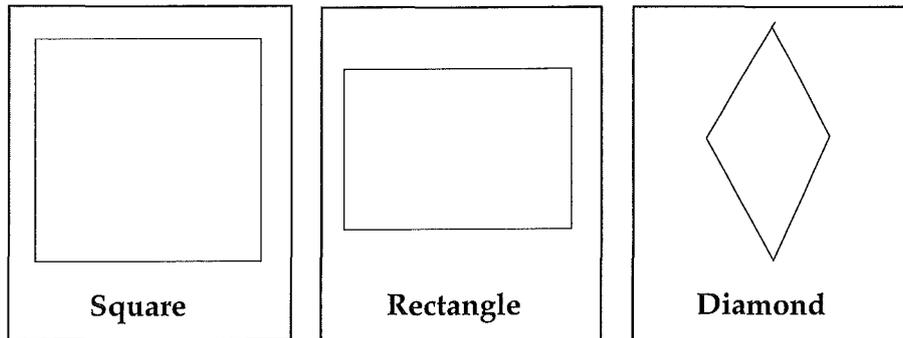


Figure 4.1:

Consider the example in Figure 4.1. All three shapes, the square, rectangle and diamond, have the same number of sides (four). All the sides for each of the objects are also a straight lines. Statistics concerning both of these features would identify all three shapes as the same. Further statistical features about the angles would help to distinguish the diamond from the square and the rectangle. For example the average angle for all three shapes would be 90 degrees. However the variance of the angles for the diamond would be more than the square and the rectangle. Furthermore, we can use length information to distinguish the square from the rectangle.

Statistical information can not guarantee uniqueness between two dissimilar shape's representations. Therefore, used independently, it will not always be accurate. Combining multiple statistics will improve this. In this case the weight each statistic carries towards the overall comparison becomes an issue. This weighting should reflect the importance of the particular statistic towards the distinguishing features of the shape.

Statistical information could also be used to eliminate some database items; although it may not guarantee uniqueness, it does offer invariance. For example two similar shapes should always produce similar statistics.

For further information on the types of statistical information used see Chapter 5.

Chapter 5

Implementation

The project implemented both extracting software and software that allowed images to be used as queries on a sample image database. This chapter describes these programs.

5.1 Extracting Software

Using our extracting software an image can be decomposed into our representation. Given a shape it extracts the boundary, which it then decomposes into our representation. The problem of extracting the edges is trivial as only simple black and white images have been used. The first stage of converting the shape's boundary into our representation is to find all the corner points. Then the segments between the corners are analysed to determine their properties or whether there are additional corner points that have not been picked up. The next step is to calculate the angles and normalise the lengths. The final step converts these into the string representation for storage.

5.1.1 Corner Detection

The corner points are found in two steps. Firstly an adaption of the IPAN99 corner detection algorithm [19] is used. This is a two-pass algorithm which defines a corner as a location where a triangle of specified size and opening angle can be inscribed into the boundary of a shape. The first phase finds such points. For each curve point p the detector tries to inscribe in the curve a variable triangle (p^+, p, p^-) constrained by a set of simple rules:

$$d_{min}^2 \leq |p - p^+|^2 \leq d_{max}^2$$

$$d_{min}^2 \leq |p - p^-|^2 \leq d_{max}^2$$

$$\alpha \leq \alpha_{max}$$

where $|p - p^+| = a$ is the distance between p and p^+ , $|p - p^-| = b$ the distance between p and p^- , and $\alpha \in [-\pi, \pi]$ the opening angle of the triangle which is computed as:

$$\alpha = \arccos \frac{a^2 + b^2 - c^2}{2ab}$$

Any point along the boundary line that meets these conditions is considered to be a candidate corner.

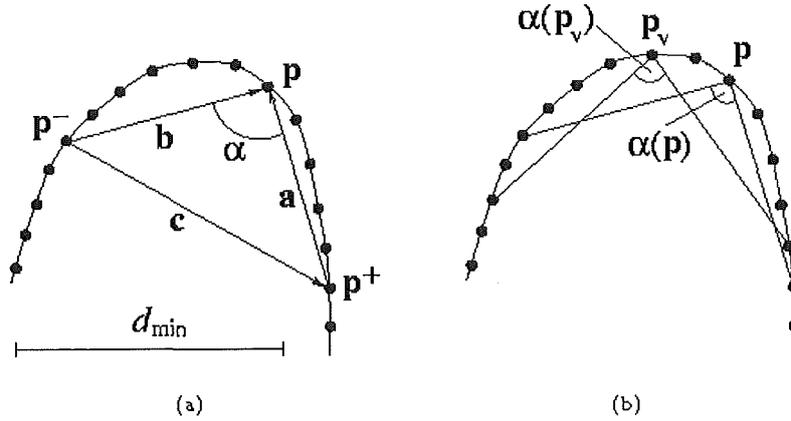


Figure 5.1: (a) Determining if p is a candidate point. (b) Testing p for sharpness.

The second phase is a post-processing step. A corner detector can respond to the same corner in a few consecutive points. A post-processing step is needed to select the strongest point by discarding the non-maxima points. A candidate point p is discarded if it has a sharper valid neighbour $p_v : \alpha(p) > \alpha(p_v)$, where p_v is a valid neighbour of p if $|p - p_v|^2 \leq d_{max}^2$. The parameters d_{min} , d_{max} and α_{max} are the parameters of the algorithm. The values we used had a strong effect on the algorithm's results. We based our values for d_{min} and d_{max} on normalised proportions with respect to the same length to which we normalised the lengths of the segments to (see section 3.3.4), and $\alpha_{max} = 150^\circ$. Figure 5.1 gives a graphical view of the algorithm.

This algorithm finds most corner points. However, it misses points of inflection that split some segments. We define a point p to be an inflection point if it is at the meeting point between a concave and a convex curve. To determine p we must scan any existing segments that we have as a result of the previous algorithm for multiple maximum and minimum points. The point p is the closest point to the midpoint of the adjoining line between any maxima and minima as shown in Figure 5.2. This step is carried out in the next phase.

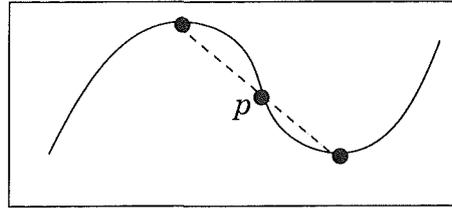


Figure 5.2:

5.1.2 Analysis Of Segments

Once the initial corner detection has been run over the boundary line, the segments between each recognised corner are analysed.

The equation for a straight line joining the two corner points is calculated. This is used to measure the variance between where the actual boundary line fits and this line. Any variation above a threshold means the the segment is not a straight line primitive.

All maxima and minima are recorded. These are the points with the most variance for the straight line joining the two corner points. Once the boundary line starts to approach the straight line between the two corner points after a maximum or minimum, and it is more than the threshold closer to the line than the last minimum or maximum, the program starts recording the next maximum or minimum in the sequence.

If there is more than one maximum or minimum between any two corner points, then these are used to calculate additional corner points using the technique illustrated in Figure 5.2. The segments between each of these new corner points is then analysed in the same manner.

If one, or no, maximum or minimum points are found, then the segment is classified as one of the three primitives. No maximum or minimum points mean that the segment is a straight line. If there is a maximum or minimum it is checked to establish whether it bends towards the internal of the object. If it bends inward it is classified as a convex curve, otherwise it is classified as a concave curve.

5.1.3 Extracting Further Information

Once all the segments and corner points have been obtained, the information about the angles and normalised segment lengths can be easily obtained. Simple geometry techniques allow us to calculate the lengths of the segments as well as the adjoining angles between any two.

The angle α at any corner point can be found with the following equation:

$$\alpha = \arccos \frac{a_1^2 + a_2^2 - l^2}{2a_1a_2}$$

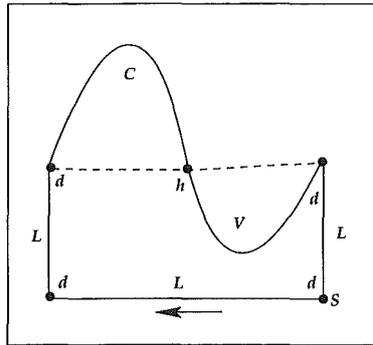


Figure 5.3:

where a_1 and a_2 and the *baseline* lengths of the two adjoining segments as shown in Figure 3.2, and l can be expressed as follows:

$$l = \sqrt{(P_1.x - P_2.x)^2 + (P_1.y - P_2.y)^2}$$

where P_1 and P_2 are the two corner points of the adjoining segments, as shown in Figure 3.2.

The a and b lengths as shown in Figure 3.1 are calculated as part of the process when analysing the segments. However, for the curves the length of the outside of the curve is also calculated for storage. This is determined by the a and b lengths of the individual curve.

5.1.4 Converting To Strings

The information extracted is stored using the string representation as discussed in [18]. Three separate strings are used, each with their own alphabet; one each for the type of segments, their lengths and adjoining angles.

The alphabet for the segments consist of three symbols; C for concave curves, V for convex curves and L for a line. The alphabets for the lengths and the angles are both of configured sizes. Since all lengths are normalised between 0 and 1, if we use sixteen symbols in our length alphabet each symbol counts for 0.0625 of the range. Likewise for the angle alphabet the range between 0 and 360 is divided into the number of configured symbols.

Consider the example in Figure 5.3. The starting point has been labelled S and the direction is indicated by the arrow pointing in the clockwise direction. The segments and angles have been marked with their corresponding symbols. The lengths have been left out for this example. The first two segments are straight lines, followed by a concave curve, then a convex curve and another straight line. The resulting segment string for this figure would be $LLCVL$. There has been assigned 16 symbols for the angles. All the angles are 90° (represented by d) apart from the one between the two curves which

is about 180° (represented by h). Traversing the angles in order the resulting string would be *dddhd*.

The extracting software, after gaining the required information, outputs the image's filename and three strings, for the segments, lengths and angles, to the database file.

5.2 Query Program

The query program takes a query image which it decomposes into the proposed representation and then into the three strings as described in the previous section. Then, for each image file in the database, it calculates its similarity with the query image. The top ranking images are kept track of and are returned in order to the user.

The query program judges the similarity using one of the three chosen techniques. The first two compare the three strings for the different shapes based on their edit distances, while the third uses statistical features.

5.2.1 Comparing Edit Distances

We have devised two techniques used for comparing two given shape's edit distances. The first uses our algorithm for finding the starting point and direction as discussed in Chapter 2. The second uses an exhaustive search of all possible starting points and directions. Both use an implementation of the algorithm shown in Chapter 4.

An important factor when comparing three strings for two given shapes is the weights each string carries and the cost of each of the three edit operations. These were parameters to our query program which the user could configure and are discussed further in the next section.

An extra factor was introduced when comparing the substitution cost between two symbols under both the length and angle alphabets. Rather than having a fixed cost if they differed, as with the segment string comparisons, we used a weighted cost depending on the variation between any two symbols. For example an A is closer to a C than it is to a J , so would be recorded as having a smaller edit distance.

5.2.2 Statistical Comparisons

The statistical comparisons extracted a series of statistical features from the stored string representations. Two shapes were compared to determine the percentage similarity they possessed under each statistical feature. Weightings were attached to represent the features importance.

There are many types of statistics that could have been used for comparisons. The features we used were:

- Counts of each type of primitive

- Means of the angle and segment length values
- Variance of the angle and segment length values
- Total number of segments
- The Total Length

5.3 Parameters and Fine Tuning

In both the extracting software and the query program there were a range of parameters whose values affected the performance of the programs.

To ensure optimal performance a variety of values were tested to determine the final values for these parameters.

5.3.1 Corner Detection

The adaptation of the IPAN99 corner detection algorithm has four main parameters that require configuration. These are d_{min} , d_{max} , α_{max} and the minimum distance between corner points (cp_{min}), as discussed previously in this Chapter. Below we give the values used for these parameters and an explanation of their effects.

- d_{min} : This was perhaps the most important parameter for our extracting software. We performed quite extensive tests to determine the optimal value. Previous work with the IPAN99 algorithm has suggested that a value of around 7 was suitable for most shapes [19], which did prove true. However, with the introduction of more variable boundary lines a larger value was needed or the algorithm identified corners that did not exist. As the lower values for d_{min} were more suitable for the smaller shapes, we decided to base the value on the size of the object. A base value of 7 was scaled logarithmically with the size of the object by multiplying it with a scale factor. The scale factor was calculated as follows:

$$scale\ factor = \frac{1}{2} \log_2 \frac{MaxDistance}{50}$$

The *MaxDistance* value is explained in section 3.3.4.

- d_{max} : Once we found a suitable value for d_{min} , we added 2 to give the value for d_{max} . This proved suitable for our program.
- α_{max} : This parameter determined whether a particular point qualified as a corner point at the first pass of the corner detection algorithm. Higher values for α_{max} meant that more corner points were found, with some of them being false. Lower values meant the opposite, that some actual corner points were classed as non-corners. After much testing we found a value of 160 degrees to be the most suitable for our test cases.

- cp_{min} : This parameter helped TO eliminate multiple corner points being predicted around the same corner. As with d_{min} , the actual value was scaled based on the size of the object. We found slightly higher values than previously recommended [19] to be the most effective. The base value we chose was 13.

5.3.2 Analysis of Segments

The only parameter used for analysing the segments related to the acceptable variance for determining maxima and minima points known as v_{min} . We selected a value of 20 for this parameter. We experimented using values based on the total length of the segment, but these had no added advantage.

This value seemed to work well for most shapes. Choosing values which are too low means that extra segments will be identified which do not accurately represent the shape, particularly when the shape's boundary is quite variable.

5.3.3 Number of Alphabet Symbols

The program also contained two parameters describing the size of the alphabets representing the lengths and angles. This was not necessary for the segment string as the alphabet size was fixed to three (one symbol for each primitive).

With the lengths, we found that the higher symbols in the alphabet were seldom used, with most segments being classified as one of the first few symbols. This meant that we had to increase our alphabet size for lengths to get a better variation between the resulting output.

We found that alphabet sizes of around 15 were reasonably effective for both lengths and angles.

5.3.4 Comparing Edit Distances

Six parameters were used to compare edit distances. The first three were the weightings that each of the three strings carried describing the segments, lengths and angles overall importance to the comparison of two given shapes. The highest importance was placed on the segment string, although the angle string also had a significant impact on the final outcome. A mixture of values was tried but no optimal values were found. The final weightings were 45% for the segment string, 30% for the angle string and 25% for the length string.

The other three parameters were the associated costs of the substitution, deletion, and insertion edit operations. Equal values were used for all three, except in the case of comparing the length and angle strings, when the substitution cost was proportional to the actual distance between the two characters.

Parameter	Value
d_{min}	$7 \times scale\ factor$
d_{max}	$d_{min} + 2$
α_{max}	160°
cp_{min}	$13 \times scale\ factor$
v_{min}	20
\sum_s	3
\sum_l	16
\sum_a	16
α_s	0.45
α_l	0.25
α_a	0.30

Table 5.1: Summary of Final Parameter Values

5.3.5 Statistical Comparisons

The parameters we used for the statistical comparisons related to the weightings that the statistical feature carried towards the final outcome. No real final values were found. This part of the program was used much less in the testing than the edit distance comparisons as it did not carry the same importance towards the final project.

Table 5.1 summarises the final values chosen for the parameters.

Chapter 6

Testing

This chapter discusses the testing of the final output from the extraction and query software. The objective is to determine the effectiveness of the proposed shape representation. To achieve this we tested the results obtained from queries on an image database containing 75 images.

Before the tests were carried out, the representation program was run on all the images in this database, acquiring the three strings for each one and storing the output in the database file. Then for each query the three shapes were also extracted from the query image. Using this information the program ranked the database images in terms of similarity to the query image using the edit distance and statistical features.

Brief testing was carried out using the statistical features and the exhaustive starting and direction method for comparing the edit distances, with most of the focus on results obtained using our algorithm for finding a unique starting point and direction.

6.1 Statistical Features

Tests showed that no single statistical feature is effective in all cases. Simple statistics such as the number of each type of primitives did, however, prove to be reasonably effective some of the time. Similar shapes did usually have similar statistics, but occasionally so too did dissimilar shapes.

Tests have shown that a correct mix of statistical features extracted from the proposed representation could be very effective if implemented well.

6.2 Exhaustive Starting Point and Direction Search

Testing showed no significant difference in the query results of the two different techniques for calculating the starting point and direction between two

different shapes. Since the technique that made use of our algorithm for finding a unique starting point and direction was less computationally expensive at query time and proved to be significantly faster, we decided to focus on that method for our evaluation and testing.

6.3 Experimental Design

Our query software was used on an image database to test the performance of the proposed representation, in terms of *precision*, *recall* and the overall *rankings*.

6.3.1 The Image Database

The experiment was carried out on an image database containing 75 images. All images were drawn by hand using a computer graphics tool. The images consisted of simple groups of similar shapes that were scaled, rotated and distorted slightly, as well as some images that were unique within the database.

6.3.2 Precision and Recall

We based our results for *precision* and *recall* on tests using 18 query images as shown in Figure 6.1. Each test ran the query program with a different image and recorded the images that it returned.

All the images returned by the program had an edit distance of less than a particular threshold with the query image. The value for this threshold was dependent on the associated costs of the three different edit operations.

For each test the values for *precision* and *recall* were measured as follows:

$$Precision = \frac{\text{Number of Similar Images Returned}}{\text{Number Of Images Returned}}$$

$$Recall = \frac{\text{Number of Similar Images Returned}}{\text{Total Number Of Similar Images}}$$

The overall values for the *precision* and *recall* were averaged over all the tests.

We also reported the precision and recall results using each of the three strings (segments, lengths and angles) individually under the same test conditions. These tests allowed us to establish the weightings for each of the three strings by assigning a greater weighting based on higher recall and precision values.

6.3.3 Ranking

For a series of tests the top matching images for each were compared with a human's interpretation for the same tests. The details of each test were the same as the *precision* and *recall* tests. For this performance measure, the order of the

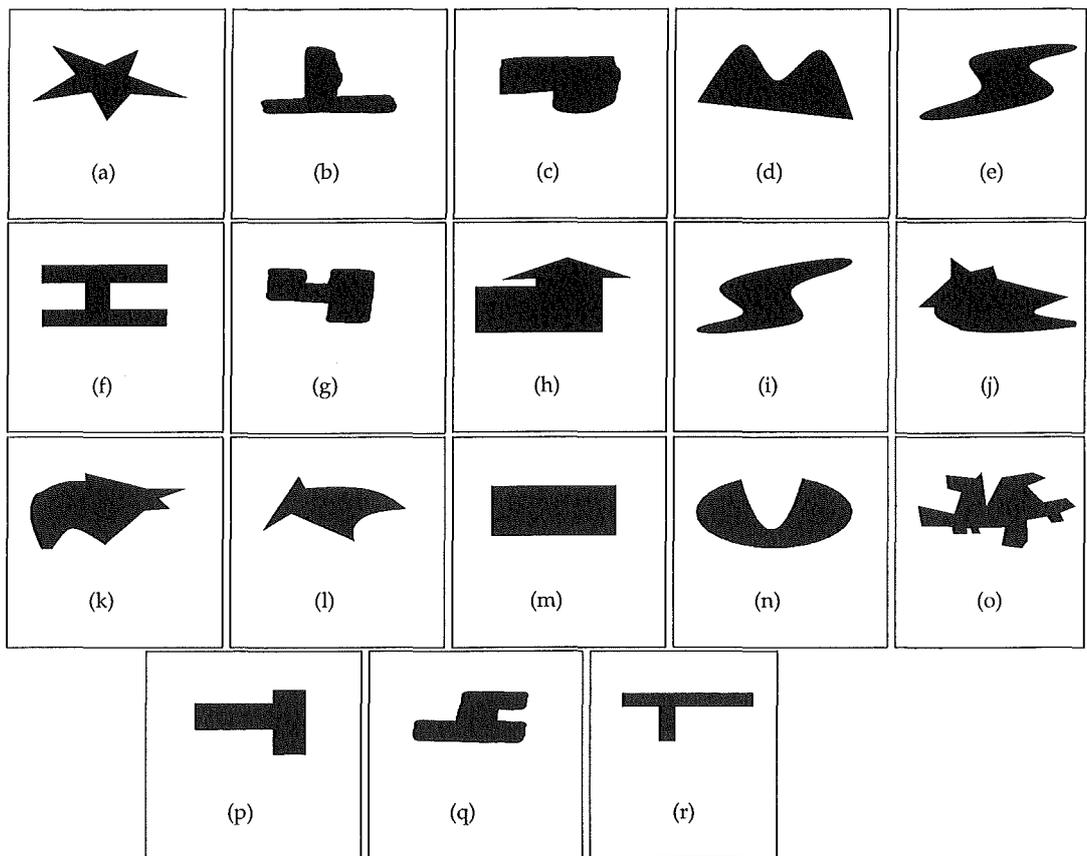


Figure 6.1: Figures Used to Test Precision and Recall

Image	Precision	Recall
(a)	0.60	0.80
(b)	0.75	1.00
(c)	0.70	1.00
(d)	0.80	0.88
(e)	1.00	0.83
(f)	1.00	1.00
(g)	0.60	1.00
(h)	1.00	0.50
(i)	0.86	0.86
(j)	0.50	0.60
(k)	1.00	0.40
(l)	1.00	1.00
(m)	1.00	1.00
(n)	0.33	0.33
(o)	1.00	1.00
(p)	1.00	0.86
(q)	0.80	1.00
(r)	0.83	0.80
\bar{x}	82%	83%
σ	20%	22%

Table 6.1: Precision and Recall Values

returned images was the most important factor. The number of ranked images compared was dependent on the number of images similar to the query image in the image database. The evaluation did not use any quantitative metric to measure the performance of the evaluation, rather it based results on a general inspection of the query program's ranking of the images with a human's ranking using the same input query.

6.4 Results

The following section reports the results of our experiments, and gives comments on how well the the representation performs on the three performance measures (*precision*, *recall* and *ranking*).

6.4.1 Precision and Recall

The proposed representation performed well for both *recall* and *precision*. Over all the tests recall averaged 82% and precision averaged 83% (see Table 6.1). Using the individual strings, the segments and lengths both had low *precision* (32% and 38% respectively). However the angles obtained a *precision* of 72%. The *recall* using the angles and lengths were both slightly low at 63% and 53%

respectively. The segment string had a much higher *recall* figure of 86%, which was higher than the total combination of all three strings.

6.4.2 Ranking

The *rankings* of the images also showed promise. For the first test (as shown in Figure 6.2), the representation performed well. The comparative rankings between the program and the human were very similar, except for the images ranked 6th and 7th, which were less similar to the query image than the 8th ranked image which appeared higher in the human's list.

There were no major differences between the human and the computer for the second test as shown in Figure 6.3. The number of similar images in the image database was much smaller for this test.

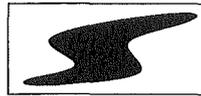
The third test (as shown in Figure 6.4) did not perform as well as the second but the representation still did well. The computer's list had images ranked at 2 and 6 that did not appear on the human's list and two similar items were ranked much lower than the rest at 12 and 14. Otherwise there was little difference.

The query shape for this test had rough edges which meant it was a harder test for the extraction program; it still performed well however.

6.4.3 Comments

All the tests showed positive feedback for the proposed representation. The segment string proved to have the most importance, but needs to be combined with the other strings to eliminate false matches. It was noted that images that did not achieve high *recall* and *precision* values produced better results using only the angles. The angles seemed to have more importance for some shapes, for example the star shown as (a) in Figure 6.1. Other situations where the representation did not perform as well can also be attributed to cases where the results from the extraction program were not optimal.

The tests definitely showed that the proposed representation has merit. Further tests against existing methods need to be conducted to establish a comparative performance.

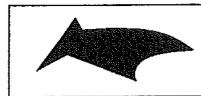


(a) Query Image

Rank	Human	Computer
1		
2		
3		
4		
5		
6		
7		
8		

(b) Ranked Images

Figure 6.2: Test One

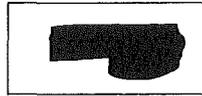


(a) Query Image

Rank	Human	Computer
1		
2		
3		

(b) Ranked Images

Figure 6.3: Test Two



(a) Query Image

Rank	Human	Computer
1		
2		
3		
4		
5		
6		
12		
14		

(b) Ranked Images

Figure 6.4: Test Three

Chapter 7

Conclusion

This project proposed a new representation for the purpose of shape based image retrieval. It defined the properties of this representation and discussed the issues concerning these. The proposed representation made use of three primitives to represent segments of a shape's boundary. The three primitives were a straight line, and concave and convex curves.

The representation was converted into three strings for comparison with other objects. The strings represented the different types of segments, the lengths of the segments and the adjoining angles between any two segments. To determine the similarity between any two shapes we used the edit distance between their three strings, using the deletion, insertion and substitution edit operations.

Software was implemented that extracted the features from a given shape, composed it into the proposed representation, and allowed ranked queries to be performed on an image database.

Testing was undertaken with this software to evaluate the proposed representation in terms of *precision*, *recall* and *ranking*. The representation achieved 82% *precision* and 83% *recall*. To evaluate the *rankings*, the computer ranking was compared against a human's ranking. The proposed representation also performed well under this measure.

7.1 Future Work

As work on the proposed representation is confined to this project, there is much room for extensions. Such work includes a more complete testing of the representation and a comparative study with other methods.

The efficiency of the software components used for this project, not a major development concern, and could be improved.

There are also many possibilities to develop the shape extraction techniques. Better techniques would make the proposed representation more practical in recognising real life images.

Another area of interest is the theoretical analysis of the properties of the proposed shape representation and its performance.

Bibliography

- [1] B Funt and G Finlayson. Color constancy color indexing. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 17(5):522–529, 1995.
- [2] M Stricker and A Dimai. Color indexing with weak spatial constraints. *SPIE conference*, 1996.
- [3] M Stricker and M Swain. The capacity of color histogram indexing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 704–708, 1994.
- [4] M Swain and D Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
- [5] RC Gonzalez and RE Woods. *Digital Image Processing*. Addison Wesley, 1992.
- [6] J Weszka, C Dyer, and A Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Transactions On Systems, Man and Cybernetics*, 6(4):269–85, 1976.
- [7] R Mehrotra and JE Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, pages 57–62, 1995.
- [8] E Milios and G Petrakis. Shape retrieval based on dynamic programming. *IEEE Transactions On Image Processing*, 9(1):141–147, 2000.
- [9] S Matusiak, D Mohamed, B Thierry, and A Olivier. Sketch-based images database retrieval. *MIS'98, LNCS 1508*, pages 185–191, 1998.
- [10] M Bouet and Khenchaf A. Fourier theory applied to shape modeling. In *Proceedings, 5th Joint Conference on Information Sciences (JCIS2000)*, pages 608–611, 2000.
- [11] M Daoudi, S Matusiak, and R Thami. Visual image retrieval by multi-scale description of user sketches. In *Proceedings, 5th Joint Conference on Information Sciences (JCIS2000)*, pages 504–508, 2000.

- [12] A Martinez-Smith, J Wang, R Sridhar, and R Acharya. Efficient shape and color based methods and architecture for content-based visual retrieval. In *Proceedings, 5th Joint Conference on Information Sciences (JCIS2000)*, pages 509–512, 2000.
- [13] S Shirani, A Jerbi, F Kossentini, R Ward, and J Wu. A shape descriptor and its application in content-based retrieval. In *Proceedings, 5th Joint Conference on Information Sciences (JCIS2000)*, pages 176–179, 2000.
- [14] M Hoffman and E Wong. Content-based image retrieval by scale-space object boundary shape representation. In *Storage and Retrieval for Image and Video Databases*, pages 86–97, 2000.
- [15] V Ogle and M Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, 1 1995.
- [16] V Gudivada and V Raghavan. Content-based image retrieval systems. *IEEE Computer*, 28(9):18–32, 1995.
- [17] J Wu, B Narasimhalu, C Lam, and Y Gao. Core: a content-based retrieval engine for multimedia information systems. *Multimedia Systems*, pages 25–41, 1995.
- [18] D Adjeroh, M Lee, and I King. A distance measure for video sequences. *Computer Vision and Image Understanding*, 75(1):25–45, 7 1999.
- [19] D Chetverikov and Z Szabo. Detection of high curvature points in planar curves. In *23rd Workshop of the Austrain Pattern Recognition Group*, pages 175–184, 1999.