A Retrieval Technique for Similar Shapes

H. V. Jagadish

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

We propose an organization for a database of objects that permits the efficient retrieval of all objects in the database with a shape similar to a search template. The retrieval technique we propose is robust in the presence of noise, and can handle several different notions of similarity including changes in scale, position, and even relative sizes of components. We can thus have the computer reproduce, on a large database of images, the process performed by a human in "riffling" through a book, using an index structure to retrieve likely candidates quickly.

1. INTRODUCTION

Given a large set of objects (or records), selecting a (small) subset that meets certain specified criteria is a central problem in databases. For records with alphanumeric fields, index structures such as B-trees are well-understood and widely used in commercial products today. For geometric or spatial objects, these index structures do not directly apply. However, there is currently much excellent work in devising novel index structures to retrieve geometric objects that intersect a specified spatial extent [6, 7, 12, 13, 15]. These structures are useful to locate objects in a spatial database that lie within (or intersect) a specified coordinate range, but are not appropriate for retrieving objects shaped like a given search shape. In other words, standard spatial indexing techniques deal with spatial location rather than with shape similarity.

Shape matching is an important image processing operation. Considerable work has been done on this problem, with different techniques being used to identify shapes, usually in terms of boundary information or other local features (*cf.* [1, 16]). Existing techniques for shape matching are *model-driven*, in that given a shape to be matched, it has to be compared individually against each shape in the database, or at least against a large number of clusters of features.

Our goal in this paper is to devise a *data-driven* technique. In other words, we wish to construct an index structure on the data such that given a template shape, matching shapes can be retrieved in time that is less than linear in the size of the database, that is, by means of an indexed look-up.

There are two existing streams of work in this direction. One technique is to index an image after having analyzed it and recognized its semantic components (*cf.* [3]). Such techniques are not applicable where an image is given with no indication of the associated semantics, since performing pattern recognition to associate semantics with an image is known to be a hard problem and is a central concern of the computer vision community. The other stream of related work is by Grosky and Mehrotra (*cf.* [5]). Their idea is to compute properties of local boundary features of objects, and then to index these. Their technique relies on small features and hence is not robust. It also requires a special index structure to be constructed and maintained, with a large fraction of the index being traversed on each access. Our work here overcomes these shortcomings by permitting the use of any standard multi-dimensional index structure. However, the price we pay is that we cannot deal with objects that may be partially occluded (or overlapping).

Finally, in addition to the goals of the related work discussed above, we have one additional goal. Not only do we wish to retrieve shapes that match the given query shape exactly, we are also interested in shapes that are "similar" to the query shape. The question then is what is similarity? One can conceive of many different dimensions along which one can measure similarity. (See [10] for an eloquent treatment of this subject). One measure that is clearly of value is "area difference". That is, two shapes are similar if the error area (where the two do not match) is small when one shape is placed "on top of" the other. In a digital domain, we obtain a pixel-wise exclusive OR of the two shapes, and pronounce the two shapes similar if the number of pixels ON in the result is small.

In this paper, we use this area-difference notion of similarity (and also some extensions of it) and show how to organize the data and construct an index to retrieve shapes in a large database that are similar to a given query shape.

Section 2 introduces the notion of a rectangular cover for a shape, and establishes the conventions we use in specifying such covers. The basic data structure required by our technique is described in Sec. 3, along with the procedures for performing several types of shape similarity queries. The issues associated with an approximate rather than an exact match are discussed in Sec. 4. We also present

experimental results obtained from a synthetic database. We present a couple of extensions in Section 5 and conclude with some final comments in Section 6.

2. SHAPE REPRESENTATION

We restrict ourselves to rectilinear shapes in two dimensions, that is, to polygons, not necessarily convex, all of whose angles are right angles so that all edges are horizontal or vertical. Since any general shape can be approximated by a fine enough rectilinear "staircase", and since digitization produces this effect in any case, we believe that this restriction to rectilinear shapes is not too limiting. We study the two-dimensional case for its ease of exposition, and because it is by far the most important case in practice. Extensions to higher dimensions are discussed in Sec. 5.1.

Rectangular covers for two-dimensional (rectilinear) shapes have been studied extensively (*cf.* [2, 4]). In this paper, we shall primarily be concerned with two types of rectangular covers. *Additive* rectangular covers are what we think of naturally: the given rectilinear shape is obtained as the union of several rectangles. *General* rectangular covers permit both addition and subtraction of rectangles, with subtraction treated as a pixel-wise set difference.

The benefit of a general rectangular cover is the possibility of considerably more succinct descriptions, as can be seen in Fig. 1. The drawback is that the process of obtaining good descriptions becomes more complex, as we shall see below.

Let C_i, with integer $i \geq 0$, be the *current* (partial) cover after i rectangles have been included in the (prospective) cover. C_0 is the empty set (of pixels or points in the plane). In an additive rectangular cover, $C_{j+1} = C_j \cup R_{j+1}$, where R_j is the j^{th} rectangle added. In a general rectangular cover, either $C_{j+1} = C_j \cup R_{j+1}$ or $C_{j+1} = C_j - R_{j+1}$, depending on whether the new rectangle is added or subtracted.

Call the shape to be covered S. For every finite rectilinear shape there exists an integer K such that we can find a $C_K = S$. No further rectangles need be added to C_K, so we define $C_j = C_K$ for $j \geq K$.

Neither the additive rectangular cover nor the general rectangular cover for a given shape is unique. Fig. 2 shows some different ways that an L shaped object could be covered additively. Clearly, we prefer the covers shown in Figs. 2b-e to the cover shown in Fig. 2a: the latter has an unnecessarily large number of rectangles in it. Even if we restrict ourselves to covers comprising exactly two rectangles, we still have many choices, even for as simple a shape as an L, as we can see from Figs. 2b-e. By convention, we shall not permit any rectangles in a rectangular cover that are "larger than necessary". Thus in the L-shape example, Figs. 2d and 2e are both disallowed, while Figs. 2b and 2c are both permitted. We define this notion formally below. Which of Figs. 2b and 2c is used depends on other requirements that may determine the order in which the two arms of the L are to be added, with Fig. 2b being selected if the horizontal arm is added first, and Fig. 2c if the vertical arm is.

When a rectangle R is added to the current additive rectangular cover C_i, there must not exist a rectangle R' contained in R ($R' \subset R$) such that $R \subseteq R' \cup C_i$ (that is, such that $R' \cup C_i = R \cup C_i$). Note that thus preventing rectangles from being larger than necessary is not the same thing as saying there should be no overlap. In particular, an additive rectangular cover for a "cross" is simply two rectangles that overlap in middle of the cross.

The same "not larger than necessary" rule applies to general rectangular covers as well. When a rectangle R is added to the current general rectangular cover C_i, there must not exist a rectangle R' that is contained in R ($R' \subset R$) such that $R' \subseteq R \cup S \cup C_i$. When a rectangle R is subtracted from the current cover C_i, there must not exist a rectangle R' that is contained in R ($R' \subset R$) such that $(R - R') \cap C_i = \emptyset$.

In [8] it has been suggested that it may be possible to describe the features of an object "sequentially" so that the most important features are described first, and any truncation of the sequence is a "good" approximation of the shape. A description thus comprises a sequence of "units" of description, where each unit iteratively refines the information provided thus far. A "Cumulative Error Criterion" has been defined to identify the best possible sequential description. According to this criterion, the error after the first unit of description, after two units of description, and so on, is accumulated, until the complete description is obtained. Thus, the error in the last stages is counted many times, while the error in the first stages is counted only a few times, and there is an incentive to minimize the error early. A general technique has been provided to find the best sequential description of a given shape.

This idea of sequential description applies in particular to rectangular covers. Our notion, in this paper, is to obtain such a sequential description of an image and then truncate it to obtain an approximate description. The claim is that this approximation leaves out the less essential features of the image, and is likely to have a small error given the criterion used to obtain the sequential description in the first place. Moreover, the truncation is likely to get rid of high frequency noise, such as specks of dirt, and other low area artifacts.

For the purposes of this paper, the specific algorithm used to obtain a good sequential description is immaterial as long as one has been agreed upon. As far as we are concerned, each shape in our database comprises an (ordered) set of rectangles (along with a positive or negative sign, if we use general rather than additive rectangular covers). The shape is described by means of the relative positions of these rectangles. In the next section we describe a storage structure for such shapes, and show how an index structure may be constructed for matching shapes.

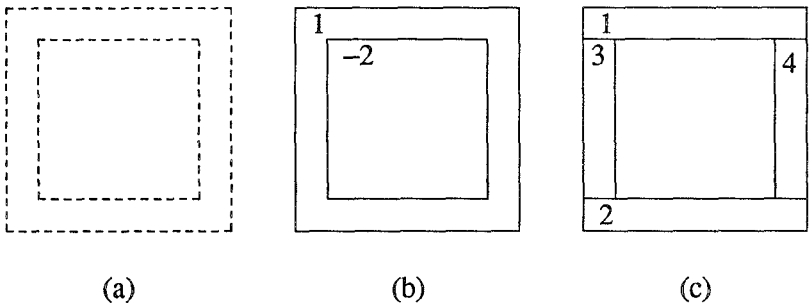

Figure 1. (a) An annular shape, (b) A General Rectangular Cover for it, and (c) An Additive Rectangular Cover for it

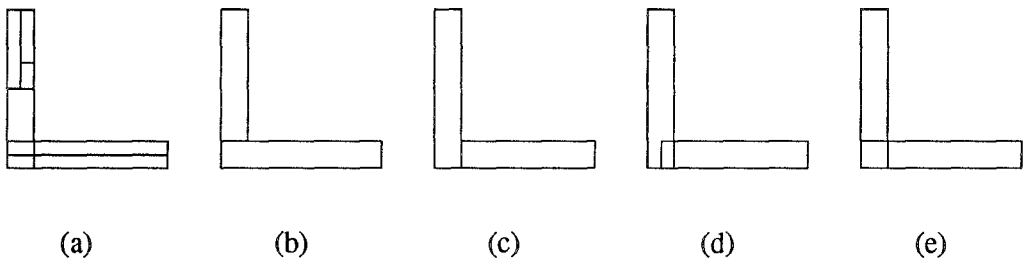

Figure 2. Some Potential Additive Rectangular Covers for an L Shape

3. RETRIEVAL OF MATCHING SHAPES

3.1 Storage Structure

For each rectangle one can identify a lower-left and an upper-right corner, which we shall call the L and U corner respectively. Each corner can be represented by a pair of X,Y coordinates in an appropriate coordinate system, such as position on the digitizing camera or screen pixels. Thus a set of K rectangles can be represented by a set of $4K$ coordinates (K rectangles times 2 corners each times 2 coordinates per corner).

To aid in the retrievals that we intend to perform, rather than store these coordinates directly, we apply a few transformations to them. First, rather than store the L and U corner points directly for each rectangle, we obtain distinct position and size values. The position of the rectangle is given in terms of the mean of the L and U corner points, i.e., the point $(\frac{x_L+x_U}{2}, \frac{y_L+y_U}{2})$. Here x_L is the X coordinate of the L corner point, and so forth. The size of the rectangle is obtained as the difference between the L and U corner points, i.e., as the pair (x_U-x_L, y_U-y_L). Thus, we still have four values, or two pairs of numbers, to store for each rectangle. However, after this transformation they represent the position and size of the rectangle rather than the locations of the corner points.

Second, the position of the first rectangle is used to normalize the positions of the other rectangles. That is, the center of the first rectangle is placed at the origin, and all coordinates are taken with respect to this origin. This transformation is represented by a shift, which is a pair of constants that has to be subtracted from all the X coordinates and all the Y coordinates respectively of the position values for each of the rectangles. Their size values remain unaffected. Since the center position of the first rectangle is 0,0 after the shift, we do not store it. Instead, we store the amount of the shift, which is given by the coordinates of this center point before the shift.

Third, the size of the first rectangle is used to normalize the positions and sizes of the other rectangles. For this, the X and Y size parameters of the first rectangle are used to divide the X and Y (both size and position) parameters respectively of all the other rectangles. (Note that the X and Y size parameters of the first rectangle are both strictly positive, and therefore can safely be used as divisors for this normalization). Further, we take the (natural) logarithms of the normalized size values, thus making them "additive" like the position values. (No logs are required for the position values). After the normalization, the size of the first rectangle is 1,1 (and its logarithm is 0,0). Rather than store this value, we store the original size parameters of this rectangle, obtained after the first transformation, which were used as the constants for this third transformation. This pair of constants are scale factors in the X and Y dimensions for the other rectangles.

Finally, we make one additional change. Rather than retain two global scale factors, one for each dimension, we retain their product as "the scale factor" (this is the square of a linear scale factor, and is an area scale factor), and their ratio (the Y scale factor divided by the X scale factor) as a "distortion factor".

Thus, a shape, described by a set of K rectangles, can be stored as a pair of shift factors for the X and Y, a scale

factor, and a distortion factor, all of which are stored as "part of" the first rectangle, and a pair of X and Y coordinates for the center point and a pair of X and Y size values for each of the remaining $K-1$ rectangles, after shifting and scaling.

The value of K, the number of rectangles required to describe a shape, could be very large for some shapes. It may not be practical to construct index structures for attribute spaces with such high dimensionality. However, we are guaranteed that "most" of the interesting shape information will be in the first few rectangles. Moreover, the basic requirement on indexing in a database is that it provide sufficient discrimination to prevent the retrieval of a large fraction of the database, and not that it produce only the exact match. This is especially true when dealing with a similarity match rather than an exact match. So it suffices to index on a small number, k, of rectangles. Our experience, in trying out various synthetic shapes, appears to indicate that a value of k, the number of rectangles indexed, of 2 to 5 suffices to provide only a few hits in a large database, even if K is an order of magnitude larger for many shapes in the database.

The shape description has, by this means, been converted into a set of coordinates for a point in $4k$-dimensional space. We can now use any multi-dimensional point indexing method that we desire, such as grid-files [11], k-D-B trees [13], buddy trees [14], holey-brick trees [9], z-curves [12], etc.

3.2 Queries

We consider four different types of queries on a database structured as described above. These are:

i. Full match
ii. Match with Shift
iii. Match with Uniform Scaling
iv. Match with Independent Scaling

We describe each of these below in turn.

3.2.1 Full Match

A full match for a given query shape is a database shape that has the correct shape in the correct position. This is the sort of thing humans do very well when riffling through the pages of a book for a particular page that we "remember". Such a match would be used, for example, if there is a large set of images, one of which has been reproduced and we now have to determine which.

To perform a full match, the query shape is transformed in the same way as each data shape has been, described in Section 3.1 above. We thus obtain a query point, and this point can be used as a key in an index search, which locates data points that are in the vicinity.

3.2.2 Match with Shift

Usually, when we think of what a shape looks like, we do not care about the position of the shape in any coordinate system. As such, we would like to retrieve similar shapes from the database irrespective of their positions in the coordinate system used to describe them. We can achieve this result as follows: transform the given query shape into a point as discussed above. Then "throw away" the two shift factor coordinates of the point, and make the query region an infinite rectangle around the point, permitting any value whatsoever for the shift factor. The relative position coordinates of the centers of all rectangles other than the first are invariant to any shifting of the entire rectilinear shape as a whole. Similarly, the scale and distortion factors are independent of any shifting. The query region obtained as above is can then be used as a key in an index search, which retrieves data points that match in all dimensions except for the two shift factors. (Since the key in the search leaves these unspecified, all values of shift factors will be retrieved).

3.2.3 Match with Uniform Scaling

Often, besides not caring about the position of the shape, we may not care about the size either. For example, the size may depend on how far the shape was from the camera, or what scale factor is used for the representation. In such a case, we can throw out the scale factor in addition to the two shift factors, and perform a retrieval as described above.

3.2.4 Match with Independent Scaling

Occasionally we may wish to permit independent scaling along the X and Y axes, rather than the uniform scaling that we normally expect. Such scaling may occur, for example, if a picture is taken at an angle to the shape. Retrieval with such a match can be performed by transforming the query shape into a point as in all the previous cases, and then using infinite ranges rather than fixed coordinate values for not just the shift factors and the scale factor, but for the distortion factor as well.

4. APPROXIMATE MATCH

In the previous section, we described the basic data structure and technique to retrieve shapes that match a given query shape in any one of several ways. The retrievals described there would each perform an exact match on the relative position and sizes of the first k rectangles in the object description (and also the distortion factor, scale factor, and shift factors, unless these have explicitly been ignored in the query). In this section we discuss the issues involved in permitting an approximate match: permitting the retrieval of shapes that are similar, though not necessarily identical, to the query shape.

4.1 Approximation Parameters

The obvious way to obtain objects of similar shape is to retrieve all objects whose shape descriptions have rectangles with similar, even if not identical, position and size as the query shape description. The way to do this is to "blur" the query point, by specifying a range along each attribute axis, corresponding to some flexibility with regard to the exact values for the position and size of each rectangle. The extent of this blurring can be determined independently for each attribute axis, by means of appropriate parameters. The larger the amount of blurring permitted, the weaker the search criterion, and the larger the set of objects selected as

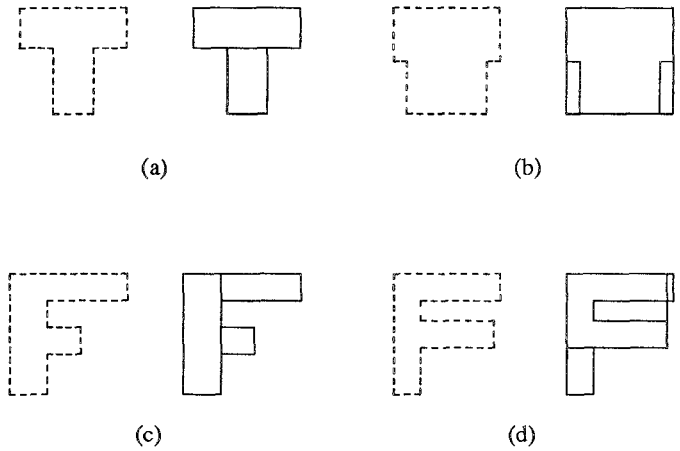

Figure 3. Similar shapes may have very different optimal (general rectangular cover) descriptions

being "similar" to the given query shape. In most applications, rather than specify independent parameters for the blur permitted in the position and size of each rectangle, global parameters can be specified. These global parameters can, for example, define the blur permitted to be an affine function of the value. Thus larger position or size values will have proportionately larger error margins allowed, but with some error margin allowed even for small values.

Given that the shape descriptions being used are sequential rather than arbitrary, a more subtle way to obtain approximation is not to use all the rectangles in the description of the query shape. Since most of the key features of the shape are expected to have been defined in the first few rectangles, similar, but not identical shapes can be expected to differ only in the last few rectangles of their descriptions. Thus by controlling the number of rectangles of description used, one can choose how strongly a shape must be similar to a given query shape for it to be retrieved. In the extreme case, for example, one could use only the first rectangle, so that a match with uniform scaling, say, would retrieve all shapes the largest part of whose mass was proportioned (height to width) in roughly the same ratio as the query shape.

For any given query, the number of rectangles to include in the search is a parameter that must be selected carefully. Clearly, if the index structure in the shape database has been constructed on k rectangles, k is an upper limit on this parameter. To be more generous in interpreting similarity, we may wish to index based on fewer rectangles. However, if too few rectangles are used, then the retrieval may return shapes completely dissimilar to the query shape. One reasonable heuristic is to truncate the description when the error area becomes a small enough fraction of he total. Another heuristic is to truncate the description when the size of error fixed by (or the size of) the next rectangle becomes a small enough fraction of the total area. Such heuristics are often reasonable, but one can always find cases where they are inappropriate. In fact, for a general (not additive) rectangular cover, it is even possible for the error not to decrease monotonically!

4.2 Multiple Representations

One potential problem with the similarity retrieval as suggested here is that some shapes may have two or more dissimilar sequential descriptions that are almost equally good, or equivalently that two fairly similar shapes may have very different sequential descriptions.

In Fig. 3, the optimal (sequential general rectangular cover) descriptions are presented for two familiar shapes (T and F). Observe how the optimal description changes as the relative sizes of the parts is changed. In both cases, there is some threshold where the switch-over occurs from one description to the other. Where a mathematical criterion would place a sharp dividing line, humans may have a fuzzy transition. Two shapes close to, but on opposite sides of, this dividing line may appear quite similar to a human eye, even though their optimal sequential descriptions are completely different. For example, a human may consider the "F" in Fig. 3c quite similar to the one in Fig. 3d. However, their descriptions are completely different.

This problem occurs not just for general rectangular covers, but for additive rectangular covers as well. Recall the additive rectangular covers for an "L" shape shown in Fig. 2. The cover of Fig. 2b is preferred if the horizontal arm of the shape has greater area, and Fig. 2c if the vertical arm has greater area.

Even worse, consider an "H" shape. In any properly balanced rendering of this letter, the left and right vertical strokes are both approximately as long and approximately as thick. Which gets selected to be the first rectangle in a sequential description is a matter of chance. The error criterion is likely to be almost identical either way. This sort of problem with multiple equally good descriptions almost always arises for symmetric shapes. The same (or almost the same) shape with two different choices of sequential description will map to two completely different points in

the attribute space over which we index.

One way to resolve this problem if two or more sequential descriptions are almost as good is to keep all of them. Thus, unless the "T" shape is really skinny and definitely a "T", its sequential description may be stored both ways. Similarly, for an "H" shape, two sequential descriptions may be stored, with one having the left vertical stroke as the first rectangle and the right vertical stroke as the second, and the other having the two in reversed order. While this approach does solve the problem, it has the disadvantage of multiplying the size of the database. In the worst case, the number of different "reasonable" sequential descriptions could be exponential in the length of the description.

A better solution is to obtain multiple "good" sequential descriptions of the given query shape and then to perform a query on each of them, taking the union of the results obtained. This way, a little more effort is required at query time, but the database and index structure do not have to expand. Moreover, the number of different sequential descriptions that have to be tried is exponential in the length of the query description which is likely to be considerably shorter than the length of the full sequential description for the query shape (and for objects in the database). In practice, this number is typically smaller than this already acceptable worst case.

4.3 Dimension Mismatch

In general, the length of the sequential description of an object need not match the number of dimensions in the index. If an object in the database has a longer sequential description, only the first part of it is used in the index structure. This will usually be the case.

Consider, however, the case where there is an object in the database with a very short description. For example, there may even be a pure rectangular shape, which requires exactly one rectangle to describe it. For such shapes with a K (the number of rectangles in the sequential description) smaller than k (the number of rectangles over which an index structure is to be constructed) we have a problem because some of the rectangles over which the index structure is to be constructed do not exist.

This problem is solved by adding to such a description $k-K$ dummy "rectangles", all with one size parameter zero, but with the other size parameter and the position parameters (in the X and Y) that represent a range from $-\infty$ to ∞ instead of a single value. Thus, these objects become hyper-rectangles in attribute space rather than simple points. Most multi-dimensional index structures can handle such hyper-rectangles, in addition to points. Since there are two size parameters, there are two choices for each rectangle and 2^{k-K} choices for the sequence of $k-K$ dummy rectangles. Thus, 2^{k-K} entries, one corresponding to each possible choice, are required for such an object.

In the case of an exact match, this scheme works in a straightforward way. In the case of a similarity match, consider a query shape that has a few additional rectangles. If the query shape is similar to the object of concern in the database (with a short description), these additional rectangles must have a small area, and therefore for each additional rectangle at least one of the two size parameters must be small. Once "blurring" is introduced for similarity retrieval, the small size parameter of each additional rectangle maps to a range that includes zero. Irrespective of the position parameter and the other size parameter of these rectangles, they will intersect appropriate dummy rectangles in (at least one of the multiple representations of) the object of concern in the database.

Conversely, if a particularly simple query shape is supplied, the above procedure can be applied to extend the query shape description. However, we also have the alternative of simply ignoring the additional attribute axes, at the cost of potentially retrieving too many objects from the database.

As mentioned above, each object with too short a description requires multiple entries in the database with the number of entries being exponential in $k-K$. This may be acceptable, if $k-K$ is small (say, 1), and there are only a few objects with such short descriptions, which is usually the case. Otherwise, the database can be redesigned as follows:

Rather than have independent size values in the X and Y dimensions, we could obtain a single size value for each rectangle as the product of the two values. This number gives the area of that rectangle relative to the first rectangle. We also obtain the ratio of the X and Y sizes, giving the distortion of each rectangle relative to the first. Now, an object with a short description need only have the single size parameter set to zero for the dummy "rectangles" in the description, and the distortion parameter set to an infinite range. Thus each objects requires a single entry in the database. The reason we have not selected this alternative is that after two ratios are taken, the value of the distortion parameter becomes sensitive to minor changes, and hence less useful as a metric for shape similarity. See Sec. 5.3 for a discussion of sensitivity issues.

4.4 An Example

Consider the shapes, representing Hindu numerals, shown in Fig. 4. For each shape, the first four rectangles in a general rectangle cover are shown below it. We have chosen to show the first four rectangles since the fifth rectangle onwards their sizes were considerably smaller than those shown. (The only exception is the '5', for which the fifth rectangle was not much smaller than the fourth, and hence is shown dashed). For all the cases, all first four rectangles were added (no subtraction until the fifth rectangle). Ignoring the global shift and scale parameters (which are roughly identical for all these shapes), the global distortion parameter and the attributes (two position values, two size values) of the second, third, and fourth rectangles are given for each shape.

Note that the numeral '3' in Fig. 4b and the numeral '5' in Fig. 4d are very similar, in fact being identical over the lower half and the top bar. As a consequence, three of the first four rectangles are identical for the rectangular cover descriptions of the two numerals.

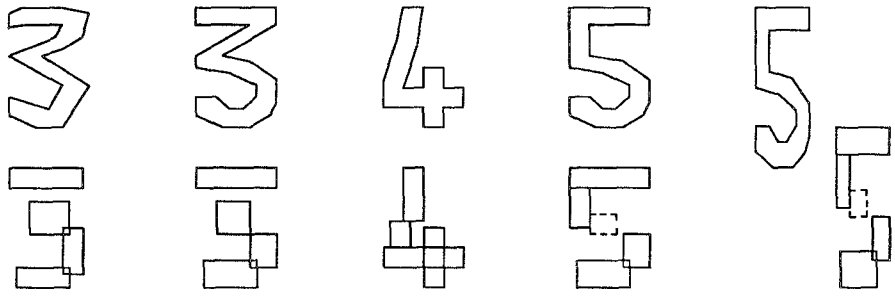

Figure 4. Some Numerals and the First Few Rectangles in Their Shape Descriptions

(a) [0.25] (+0.05,-2.00,-0.60,+0.51) (-0.05,-5.00,-0.31,+0.00) (+0.36,-3.50,-1.31,+0.85)

(b) [0.25] (-0.08,-4.83,-0.40,+0.29) (-0.04,-2.00,-0.87,+0.51) (+0.33,-3.67,-1.11,+0.51)

(c) [0.25] (+0.13,+0.00,-1.39,+1.10) (-0.13,+3.17,-1.39,+0.98) (-0.29,+1.17,-1.39,+0.29)

(d) [0.25] (-0.08,-4.83,-0.40,+0.29) (+0.33,-3.67,-1.11,+0.51) (-0.38,-1.50,-1.39,+0.69) (-0.08,-2.33,-1.11,+0.00)

(e) [0.50] (-0.08,-4.83,-0.40,+0.29) (+0.33,-3.67,-1.11,+0.51) (-0.38,-1.50,-1.39,+0.69) (-0.08,-2.33,-1.11,+0.00)

Rectangular Cover Descriptions of the Shapes Above

(a') [0.25] (-0.05,-5.00,-0.31,+0.00) (+0.05,-2.00,-0.60,+0.51) (+0.36,-3.50,-1.31,+0.85)

(a") [0.25] (-0.05,-5.00,-0.31,+0.00) (+0.36,-3.50,-1.31,+0.85) (+0.05,-2.00,-0.60,+0.51)

Modified Rectangular Cover Descriptions of the Query Shape (a)

Consider a small database comprising the three numerals in Figs. 4b-d. Suppose that the somewhat crooked numeral '3' of Fig. 4a is supplied as a query to this database. Let us see how well this query shape matches each of the three shapes in the database. Comparing the vector of values for (a) and (b) above, the distortion factors are identical, and the X positions of the second rectangles are close to zero (+0.05 and -0.08) in both cases. However, there is a large difference between the Y position of the second rectangle in (a) and in (b). Therefore we conclude that (b) is not a good match for (a). By a similar argument, we also conclude that neither (c) nor (d) is a good match for (a). However, let us recall Sec. 4.2 and try a few different representations for the given query shape. Since the areas of the biggest rectangles do not differ by very much, we could try reordering them. In this case (unlike for the 'L' shape example of Figs. 2b and 2c), the rectangles themselves are not altered when the order is permuted. (Not all permutations need be tried -- only those in which the rectangles remain in roughly decreasing order of area. The more forgiving we are in calling some order "roughly decreasing", the more the number of matches we will find of similar shapes). Two of these permutations are shown below. (The other permutations produce no matches in this database).

Now we find that the shape described in (a') is an excellent match to (b) in terms of position (maximum absolute error in any of the position parameters is 0.17 normalized units) and a moderately good match in terms of size (maximum absolute error in any of the size parameters is 0.34). Thus, we may accept (b) as a shape similar to (a), provided our threshold is low enough for accepting differences in size, and for permuting the sequential order.

The shape described in (a") is a reasonable match for (d). If only the first three rectangles are considered, with the last rectangle in (a") ignored, then we have the same maximum absolute error values as above: 0.17 units for position and 0.34 units for size. However, with the fourth rectangle included, these error values increase to 0.5 for position, and 0.79 for size. These error values are large enough that a selective enough similarity query may not report (d) as a match for (a). In other words, in spite of the numerals '3' and '5' in Figs. 4b and d being so similar, our similarity retrieval technique is able to find the '3' of Fig. 4b as being more similar to the crooked '3' query shape of Fig. 4a. It also find the '4' shape less similar to the '3' shape than the '5' shape. Both these results are exactly what we would hope for, in terms of human intuition.

Finally, suppose that the shape of Fig. 4e is specified in the query. This shape is identical to that in Fig, 4d except that it is much taller and thinner. In fact, its sequential representation is the same as that of Fig. 4d except for the change in distortion factor. A retrieval of '5' as the matching shape poses no difficulty.

4.5 Experiment

To verify the practical utility of our proposed technique a database of 16,000 synthetic shapes was constructed. Each shape was created by the amalgamation of 10 randomly generated rectangles. Sequential descriptions using additive rectangular covers were obtained for each shape, and stored in the database. Various query shapes were tried. As expected, when a shape from the database was used as the query shape, the shape itself was always retrieved in response to the query. If the error margins were small, no other shapes were retrieved. Also, as expected, if a small perturbation on a database shape was used for the query, the original database shape was still retrieved, and no other shapes were retrieved, provided the error margins were small enough, and a long enough description was used to perform the query.

A more interesting query is shown in Fig. 5. Here, an arbitrary shape, shown in Fig. 5a, was used to query the database. Not surprisingly, the space of all possible two-dimensional shapes is amazingly large, and no shapes were retrieved when the error margins were small. However, as the error margins were relaxed, we began to retrieve "similar" shapes. Figs. 5b shows the "most" similar shape in the database, retrieved using an approximate match on ten parameters: size, distortion, and four parameters for each of two rectangles. If the size parameter is dropped, and the search based on nine parameters, then the shape of Fig. 5c is also retrieved. Observe that this shape is perhaps more like the query template than Fig. 5b, but it is certainly a lot bigger. Finally, if the distortion parameter is dropped, and the search based only on eight local parameters, then Figs. 5d and 5e are retrieved as well. Observe that both these shapes are too broad, and Fig. 5d is not tall enough, to match the template without distortion. However, appropriate scaling (independently) in the two dimensions, can achieve a good match, and these shapes have been retrieved since by dropping the distortion parameter we indicated that we were willing to permit such scaling.

All the above matches were performed using the first three rectangles of the description. We next varied the length of description used in the query to observe the effects. When the description used in the query was reduced to two rectangles, so that only 4 parameters were used, with the error margins the same as before, almost a hundred shapes were retrieved. Once the error margins were tightened enough, the only shape retrieved was the one in Fig. 5f. Here the two biggest rectangles in the additive cover match the template almost perfectly. However, the shape as a whole really does not look like the template. So, in this database, a query on less than three rectangles appears too weakly constrained.

Next we tried indexing on a longer description: four instead of just three rectangles. The problem now is that the shape in Fig. 5a has only three rectangles in its description. Following the discussion of Sec. 4.3, we tried two different queries: one with the fourth (dummy) rectangle having a height of zero, and another with a width of zero. With error margins comparable to those before, only Fig, 5e was ruled out in both cases. (Figs. 5b and 5c were both accepted in the zero height case, and Fig. 5d in the zero width case). Since most of the matching shapes returned were similar whether three or four rectangles were used in the query, we may conclude that there is no need to overconstrain the query by using four-rectangle-long descriptions: the use of three rectangles is enough.

From the foregoing we have seen that the shapes returned from the database in response to an approximate match query are indeed somewhat similar to the query shape. The question that remains is whether these are indeed the "most similar" shapes in the database. This question can, of course only be answered subjectively. Since the database was too large for a human to study it, a three rectangle retrieval was performed with very loose error margins, to obtain forty shapes. These forty were then visually examined. Most of them did not resemble the given query shape at all. Figs. 5b-e were indeed judged, by one human, to be closest to the template out of the forty. The argument then is that if our technique can find the four best matches out of the forty shapes that are somewhat like the template, and hence most likely to cause confusion, then our technique must also have done a good job in selecting these four out of the 16,000. Figs. 5 g-j show four out of the forty shapes that were subjectively judged closest to the query shape, after the shapes in Fig. 5b-f (which were all also included in the forty). To the extent that you, the reader, agree that the shapes in Figs. 5g-j are less like the query shape than the ones in Figs. 5b-e, you are agreeing with the subjective evaluation described in this paragraph.

5. EXTENSIONS

5.1 Sensitivity Reduction

The storage structure as described computes the sizes and positions of all the rectangles in the description with respect to the first rectangle. Consequently, the values stored for these sizes and positions are a function of the exact size and location of the first rectangle. To the extent that the first rectangle has a large area, and is the dominant rectangle in the description of the shape, this dependence is acceptable, and even desirable. However, when there is no clear single dominating rectangle in the shape, and when the first rectangle is long and narrow (rather than almost square), small changes in the width (the smaller dimension) of this rectangle can cause disproportionately large changes in the size and position values recorded for all the other rectangles in the description. Such sensitivity may be undesirable.

A solution to this problem is to normalize all sizes and positions not with respect to the first rectangle in the description, but rather with respect to a bounding rectangle for the entire shape. In this fashion we not only remove the undesirable sensitivity, but we also have a more natural interpretation of the size and position values stored and indexed on. In addition, all rectangles, including the first, are now treated in the same fashion, potentially simplifying the code somewhat. The drawback here is that we no longer

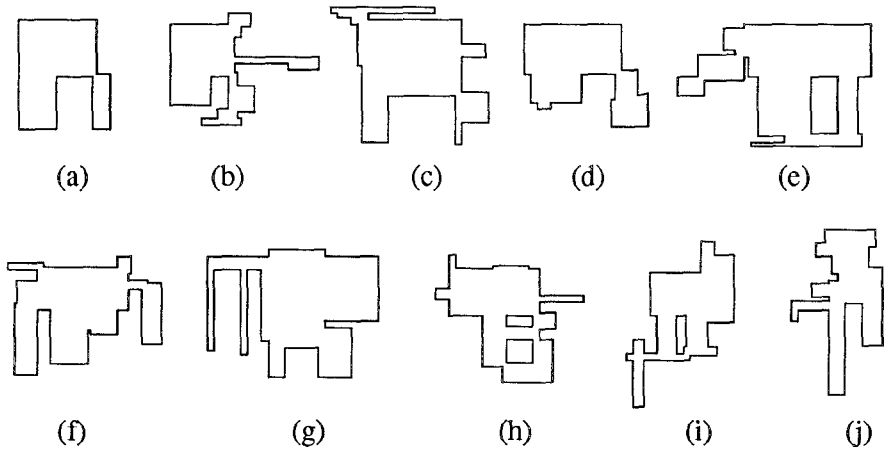

Figure 5. (a) A Query Shape, and (b-j) "Similar" Shapes in a Database

get to store the first rectangle "for free". We now have to store four parameters for each rectangle, including the first, and the global parameters in addition. Not only does this cause an increase in the storage requirement, it also increases the cost of indexing since for approximately the same selectivity, we now have to index in a space with a dimension that is higher by four than what it was before.

Thus there is a tradeoff between undesirable sensitivity and increased indexing cost. Depending on the relative cost measures, one can employ the technique suggested in this section, or stay with the technique used through the rest of this paper.

5.2 Higher Dimensions

Thus far we have described the scheme only for two-dimensional shapes. However, the techniques we propose naturally extend to multiple dimensions. For example, in the three-dimensional situation, each hyper-rectangle would be identified by means of two corner points, each point having three coordinates (one for each dimension) for a total of six attributes per hyper-rectangle. The first hyper-rectangle would be replaced by three shift factors, one scale factor, and two distortion factors (one for Y/X and one for Z/X). Each additional hyper-rectangle will have its two corner coordinates transformed into a set of three position values (one in each dimension), and a set of three size values (one in each dimension), all with respect to the first hyper-rectangle. All the ideas presented in this paper carry through with obvious small modifications.

6. DISCUSSION

In this paper, we have introduced an indexing technique to retrieve from a database shapes that are similar to a given query shape. Using this technique, we have the capability of using appropriate index structures to have a computer reproduce on a database of images the process performed by a human in "riffling" through a book. Images that appear "similar" to a search template can be retrieved and examined further.

The technique presented is simple and can be implemented easily on top of any standard multi-dimension indexing technique. In this paper, we showed how this technique can be used for an area-based similarity measure, even in the presence of scaling and/or shifting in one or all dimensions. We also showed how this technique can be used for a topological similarity measure.

The core ideas in our technique can also be used for other good continuous descriptors of shape such as moments. However, such extensions were not considered in this paper. In particular, we did not show how to handle similarity of texture (or fill patterns), since there may be no area similarity after a "phase shift". Also we do not handle rotation or occlusion, or even composition of multiple shapes, though extensions to handle some of these may be possible through the definition of appropriate signatures or moment functions as additional attributes in a shape description. Consequently, our technique is not likely to be useful in the extensively studied assembly-line robot-vision situation, where there are only a few different shapes of parts possible so that traditional methods, described in the image recognition literature, do quite well.

On the other hand, in applications with a large number of images where the orientation is largely fixed, traditional methods are likely to be grossly inefficient and our technique is likely to be of value. For example, a doctor treating a patient for a kidney stone may be able to pull-up records of past patients who had similar shaped stones from a database of medical X-rays. With some additional work, it appears likely that the technique proposed in this paper can be applied to other currently difficult applications such as criminal picture identification by the police.

This paper has considered the problem of locating similar shapes from a large database of shapes. A related problem is how to locate a particular shape in a large image. Work is currently under way to address this problem. Unfortunately, the results here do not appear to extend in any straightforward fashion!

Acknowledgements

I am grateful to Freddy Bruckstein for starting me on the line of research presented here, to Shaul Dar for a careful reading of a draft, and to Paul Krzyzanowski for his thought-provoking questions and enthusiasm.

References

[1] N. J. Ayache and O. D. Faugeras, "HYPER - A New Approach for the Recognition and Position of Two-Dimensional Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **PAMI-8**, 1986, 44-54.

[2] S-K. Chang, Y. Cheng, S. S. Iyengar, and R. L. Kashyap, "A New Method of Image Compression Using Irreducible Covers of Maximal Rectangles," *IEEE Trans. on Software Engineering*, **14**(5), May 1988 , 651-658 .

[3] S-K. Chang and S-H. Liu, "Picture Indexing and Abstraction Techniques for Pictorial Databases," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **PAMI-6**(4), July 1984, 475-483 .

[4] D. S. Franzblau, "Performance Guarantees on a Sweep-Line Heuristic for Covering Rectilinear Polygons with Rectangles," *SIAM J. Disc. Math.*, 2(3), 1989, 307-321.

[5] W. I. Grosky, P. Neo, and R. Mehrotra, "A Pictorial Index Mechanism for Model Based Matching," *Proc. Fifth IEEE Int'l Conf. on Data Engineering*, Los Angeles, CA, 1989, 180-187.

[6] A. Guttman, "R Trees: A Dynamic Index Structure for Spatial Searching," *Proc. ACM SIGMOD Int'l Conf. on the Management of Data*, 1984, 47-57.

[7] H. V. Jagadish, "Spatial Search with Polyhedra," *Proc. Sixth IEEE Int'l Conf. on Data Engineering*, Los Angeles, CA, Feb 1990.

[8] H. V. Jagadish and A. M. Bruckstein, "On Sequential Shape Descriptions," *Pattern Recognition*, 1991 (to appear).

[9] D. B. Lomet and B. Salzberg, "A Robust Multi-Attribute Search Structure," *Proc. Fifth IEEE Int'l Conf. on Data Engineering*, Los Angeles, CA, Feb. 1989, 296-304.

[10] D. Mumford , "The Problem of Robust Shape Descriptors ," Center for Intelligent Control Systems Report CICS-P-40 , Harvard University, Cambridge, Mass. , Dec. 1987 .

[11] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The Grid file: An Adaptable Symmetric Multikey File Structure," *ACM Trans. on Database Systems*, **9**(1), 1984.

[12] J. A. Orenstein and F. A. Manola, "PROBE Spatial Data Modeling and Query Processing in an Image Database Application," *IEEE Trans. Software Engg.*, **14**(5), May 1988, 611-629.

[13] J. T. Robinson, "K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indices," *Proc. ACM SIGMOD Conf. on the Management of Data*, 1981.

[14] B. Seeger and H. P. Kriegel, "The Buddy Tree: An Efficient and Robust Access Method for Spatial Database Systems," *Proc. 16th Int'l Conf on Very Large Databases*, Brisbane, Australia, Aug. 1990, 590-601.

[15] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+ Tree: A Dynamic Index for Multidimensional Objects," *Proc. 13th Int'l Conf on Very Large Databases*, Brighton, U. K., Sep. 1987, 507-518.

[16] T. P. Wallace and P. A. Wintz, "An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors," *Computer Graphics and Image Processing*, **13**, 1980, 99-126.