AN EFFICIENT SHAPE-BASED APPROACH TO IMAGE RETRIEVAL

I. FUDOS and L. PALIOS

7-99

Preprint no. 7-99/1999

Department of Computer Science
University of Ioannina
451 10 Ioannina, Greece

An Efficient Shape-based Approach to Image Retrieval

Ioannis Fudos and Leonidas Palios
Department of Computer Science
University of Ioannina
GR45110 Ioannina, Greece
{fudos,palios}@cs.uoi.gr

Technical Report 99-7
March 1999

Efficient image retrieval from large image bases is an issue central to a wide range of real world applications, such as, digital libraries, digital film databases, environmental sciences, and satellite image bases. The functionality of early image base systems relied on a preprocessing phase during which each image or group of images was manually annotated. Modern systems draw their power from sophisticated retrieval based on color, texture or other global characteristics. Such systems, although powerful and fast, often lack in accuracy. Therefore, there is an increasing demand for image retrieval techniques which will make use of local features such as shape.

Our work focuses on the problem of finding good matches for a given query shape among candidate shapes stored in a shape base. We present an efficient matching algorithm built around a novel similarity criterion. Unlike previous work which achieved an initial positioning of the query shape by normalizing one of its edges, our approach is based on the normalization of the diameter of the shape, thus making it less susceptible to errors due to noise or limited accuracy during the shape extraction procedure. Our matching algorithm works by gradually "fattening" the query shape until sufficiently many similar shapes have been discovered. The algorithm exhibits poly-logarithmic average time behavior. The user may customize the algorithm's operation by providing information regarding the desired maximum accuracy of the matching.

1 Introduction

The last few years, there is an emerging need to organize and efficiently use large pools of images that have been collected over the last decades and contain information potentially useful to areas such as medicine, journalism, weather prediction, environmental sciences, art, fashion and industry. It is estimated that there are more than 20 million pages containing hundreds of millions of images on world wide web pages alone [4]. Traditionally, images were retrieved by their filename, other technical characteristics such as date and size or, in the case of images having manual annotation, through text keywords. It was soon realized that manual annotation was not sufficient, since besides being a time consuming and not real-time process, it also describes only a very small percentage of the information that an image contains.

During the last decade, there is an increasing effort to organize and retrieve images by content based on characteristics such as color, texture and shape. There is a number of methods in the literature that perform indexing and retrieval based on global image characteristics such as color, texture, layout, or their combinations. QBIC [8, 13], a system developed at IBM Almaden supports retrieval by color histograms, texture samples (based on coarseness, contrast and directionality), and shape. QBIC uses R^* trees to process queries based on low-dimensionality features, such as, average color and texture. Shape matching is supported using either dimensionality reduction, which is sensitive to rotation, translation and scaling [16], or by clustering using nonlinear elastic matching [7], which requires a significant amount of work per shape comparison and some derived starting points as a matching guide. QBIC also supports video queries. Elastic matching based on numerical methods is a technique that has been used for shape retrieval (see e.g. [3]).

Ankerst et al [1] present a pixel-based shape similarity retrieval method that allows only minor rotation and translation. Their similarity criterion assumes a very high dimension (linear to the number of pixels in the image), therefore dimensionality reduction is performed.

Gary and Mehrotra [14, 10, 9] present a shape-based method, which stores each shape multiple times once per edge. More specifically, the shape is positioned by normalizing one of its edges. The space requirements of this method impose a significant overhead. The method is quite susceptible to noise, thus the authors present a sophisticated preprocessing phase to eliminate the noise effects. Finally, the method favors those shapes of the shape base, which have almost the same number of vertices as the query shape.

Cohen and Guibas [6] present an image retrieval method based on geometric hashing. This method calculates the hash signature of the shape based on the contributing line segments. The method has been applied to retrieve Chinese characters and is sensitive to rotation and translation. The geometric hashing

described in [6] is not related to the geometric hashing presented in our work.

In this paper, we present a shape-based method, where information regarding the boundary of objects is automatically extracted and organized to support a polylogarithmic (in the number of shape vertices) algorithm based on a novel similarity criterion. Specifically, this paper makes the following technical contributions:

- introduces a new similarity criterion for shapes, that works better in the context of image retrieval than traditional similarity criteria.
- describes a novel method of storing shapes representing the boundary of image objects; the method is noise tolerant and rotation-, translation- and scale-invariant.
- presents an efficient algorithm for finding the closest match(es) to a given query shape.
- introduces a geometric hashing technique that can be used in conjunction with the algorithm mentioned above.
- describes an interactive prototype system implementing the above methods.

The rest of this paper is organized as follows. Section 2 presents our similarity criterion for shapes and compares it with existing criteria. Section 3 describes the organization of the data describing the shapes, and the methods for their effective retrieval. Section 4 provides a description of an interactive system that incorporates our approach, and finally Section 5 offers conclusions and directions for future work.

2 Similarity Criteria

The Hausdorff distance is a well studied similarity measure between two point sets A and B. The directed Hausdorff distance h and the Hausdorff distance H are defined as follows:

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

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

where d is a point-wise measure, such as, the Euclidean distance. An inherent problem with the Hausdorff distance is that a point in A that is farthest from any point in B dominates the distance. To overcome this problem, Huttenlocher and Rucklidge have defined a generalized discrete Hausdorff distance (see e.g. [12]), given by the k-th ranked distance rather than the maximum:

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

$$h_k(A,B) = \text{kth}_{a \in A} \min_{b \in B} d(a,b)$$

This metric, eliminates somehow the farthest-point domination disadvantage of the Hausdorff metric, but works only for finite set of points, and needs to know the number of points m of the point sets to get a reasonable measure by selecting $k = m/2$. The generalized Hausdorff distance does not obey the metric properties.

An interesting alternative measure is presented in [7] called Nonlinear Elastic Matching $NEM_r(A,B)$ with stretch cost r. This measure does not obey exactly the traditional metric properties but a relaxed set of metric properties instead. In practice, this provides the same advantages as any metric, and therefore can be used for clustering. However, the arbitrary number of points distributed on the edges, the need of determining certain starting matching points and the complexity of computing such a match ($O(mn)$ using dynamic programming [2]) makes this measure inappropriate for our method.

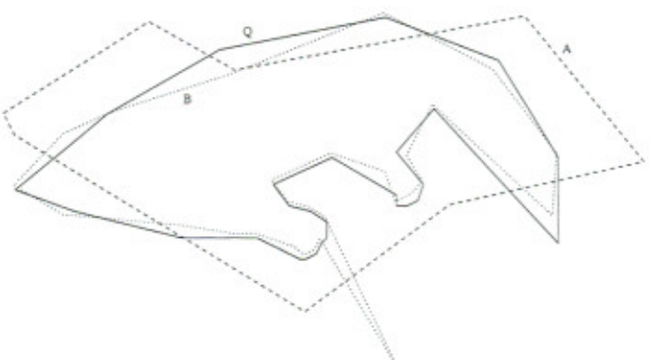


Figure 1: Depending on the similarity criterion the query shape Q may be matched with A or B

In our algorithm we use a new similarity criterion based on the average of minimum point distances:

$$h_{avg}(A,B) = \text{average}_{a \in A} \min_{b \in B} d(a,b)$$

This measure behaves nicely (gives intuitive results) and it can be computed quite efficiently, as is shown in the next section. The metric properties do not hold for this measure either, but in some sense they hold for a representative average set of points probably different from the original point set. Figure 1 illustrates an example where using Hausdorff distance the shape Q is matched

with A instead of B (B is intuitively the closest match). According to the similarity measure used in our work, B is indeed closer to Q than A.

3 Efficient Retrieval of Similar Shapes

3.1 Creating the shape database

To create the database of shapes, we process each shape element, a polygon or polyline extracted from an image, as follows. First, we compute the diameter of the shape, i.e., the pair of vertices that exhibit the longest Euclidean distance. Then, we find all the pairs of vertices whose distance is at least α times the length of the diameter ($0 < \alpha \leq 1$). For each such pair of vertices, we then scale, rotate, and translate the shape so as these two vertices get identified with the points $(0,0)$ and $(1,0)$ respectively. Each shape is stored twice for every diameter AB, once by identifying A with $(0,0)$ and B with $(1,0)$, and vice versa. All these "normalized" copies of the shape constitute the *shape base* a database of shapes.

The use of the diameter makes the retrieval system more tolerable to noise situations. During automated extraction of shapes via image processing techniques, distortion of the edges may be introduced. In Figure 2 the distorted shape (right) can not be retrieved via the query shape (left) if we store "normalized" copies of each shape based on its edges (as in [10], i.e scaling, rotating, and translating so that the selected edge is positioned on $((0,0),(1,0))$). The diameter is less susceptible to such local noise.

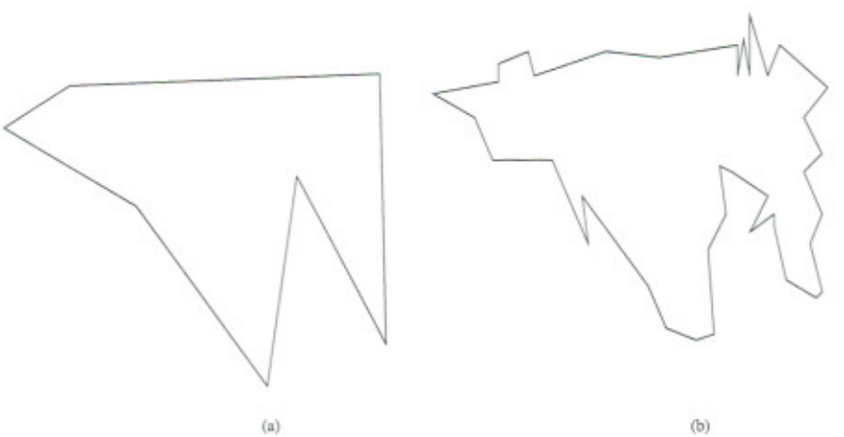


Figure 2: (a) the query shape (b) some distorted shape extracted from an image

Moreover, the use of the α ratio rule to store a number of copies of the shape in the shape base allows us to achieve even better noise tolerance. On the other

hand, this does not increase substantially the size of the database. Experiments have indicated that for $\alpha = 0.15$ the number of copies of each shape is about 14 on the average (the average number of edges per shape in the test set was about 20).

The initial source of our shape pool consists of a set of polygons or polylines that are not self-intersecting.

3.2 The matching algorithm

The algorithm relies on the computation of a "fattened" version of the query shape by taking lines parallel to its edges at distance ϵ on either side (Figure 3); we call this fat shape the *ϵ-envelope*. The ϵ-envelope can be seen as a collection of trapezoids of height 2ϵ, one for each edge of the query shape. (For simplicity, we assume that ϵ is such that no two trapezoids are overlapping; the method is easily extended to handle overlapping trapezoids.) The basic steps of the algorithm for the retrieval of the database shapes that are similar to the query shape are:

a. We compute an initial value $\epsilon = \epsilon_s$ such that the ϵ_s-envelope is likely to contain at least one shape of the shape base (see Section 3.4). We then calculate the actual number K_ϵ of vertices of the database shapes. If K_ϵ greatly differs from the estimated number of vertices that are needed (see section 3.4) then we continue, otherwise we adjust ϵ by performing binary search in the values of ϵ.

b. For each of the trapezoids T of the current ϵ-envelope, we collect the vertices of the database shapes that fall within T. This can be achieved by preprocessing the vertices so that inclusion in a query triangle can be answered fast. Available data structures of sub-quadratic space complexity allow $O(\log^3 n + \kappa)$ query time, where n is the total number of vertices of the shape base and κ is the number of vertices inside the query triangle [11]. (There are also quadratic-size data structures that allow for $O(\log n + \kappa)$ query time by using fractional cascading which uses more sophisticated linking [5].) Then, finding which among the vertices of the shapes in the database fall inside the ϵ-envelope takes $O(m \log^3 n + K)$ time in total, by processing each of the m trapezoids separately and treating each one of them as two triangles; K is the total number of vertices of the database shapes inside the ϵ-envelope. Each time we find that a certain vertex of some shape is inside a trapezoid, we increase a counter associated with that shape that holds the number of its vertices that are inside the trapezoids. By maintaining this information, at the end of this process, we know which polygons or polylines of the shape base have more than a fraction $(1 - \beta)$ of their vertices inside the ϵ-envelope, for a parameter β $(0 \leq \beta < 1)$.

c. If there are polygons or polylines of the shape base that have more than a fraction $(1-\beta)$ of their vertices inside the ϵ-envelope, we process them as described in Section 3.3; during the processing, we may either conclude that the best matches have been found in which case they are reported to the user and the execution is complete, or (i.e. we are not certain that the best match has already been found) we continue to the next step. If no candidate shapes are found a new ϵ is computed via cost calculation (Section 3.5) and we go to step e.

d. Let p_{j_1}, p_{j_2}, ..., p_{j_k} be the k polygons with the smallest k costs, s_{j_1}, s_{j_2}, ..., s_{j_k}, and s_{max} be the maximum of these costs. If we increase ϵ, then the smallest normalized cost s_{new}^{min} of a new candidate shape (that was not candidate during the last iteration) is the minimum of the cost $g_i\epsilon + t_i$, over all shapes i that were not candidate last time and therefore they had a fraction g_i, $\beta \leq g_i < 1$ of their vertices outside the envelope (t_i is the normalized cost of the vertices that were inside the envelope). If $d_{max} \leq s_{new}^{min}$, then we terminate and report the k polygons as the best matches. Otherwise, we set $\epsilon = \frac{s_{new}^{min} - t_p}{g_i}$, and go to step e. This is actually the maximum ϵ, after this no iteration is needed so we can make a binary search to find an envelope that has reasonably many vertices, which takes time logarithmic in the number of vertices on the average.

e. We check ϵ; if it is not too large, we go to step b and repeat the procedure, otherwise we report the k candidate shapes with the smallest costs (if any) and exit.

The method converges and if there exist similar shapes it retrieves the best match(es).

An alternative but fully compatible method based on geometric hashing is also outlined (Section 3.6). We revert to this method when the closest match exhibits a sufficiently large cost with respect to the query shape.

The overall time complexity after I iterations is $O(I\, m\, \text{poly-log}n + K_{\epsilon_f})$ where K_{ϵ_f} is the total number of points within the final envelope of width ϵ_f. (We note that when the width of the envelope is increased from ϵ to ϵ', we concentrate on locating the vertices of the database shapes that are inside the ϵ'-envelope and are outside the ϵ-envelope; in this way, points are not double-counted.) We expect that the number of iterations I will be logarithmic on the average, so that the total complexity is polylogarithmic in the number of vertices of the shapes in the shape base.

3.3 Processing the Polygons or Polylines

Let $S_1, S_2, ..., S_m$ be the sequence of the edges of the query shape along a clockwise traversal of its boundary starting from the vertex that was positioned at $(0,0)$ in the normalization step. Then we process each polygon or polyline P of the shape base which contains less than a fraction β of its vertices outside the current ϵ-envelope as follows (we refer to P as *candidate shape*):

(i) Starting at P's vertex which was positioned at $(0,0)$ in the normalization process, we traverse its boundary clockwise. In the general step, we have a vertex V_s inside a trapezoid of the envelope; let that trapezoid correspond to the edge S_i of the query shape (note that $(0,0)$ lies in the trapezoid of S_1). Let V_r be the next vertex of P inside the ϵ-envelope, and let S_j be the edge corresponding to the trapezoid that contains V_r.

(ii) If V_sV_r is an edge of the current candidate shape, then if the edge is completely inside the ϵ-envelope we continue to the next step. Otherwise, we check that

$$(i-2) \bmod m \leq j.$$

If this inequality does not hold, we conclude that this shape is not a candidate for successful matching and we ignore it. If not, we calculate the minimum ϵ that will bring the edge inside the ϵ-envelope.

(iii) If V_sV_r is not an edge of the candidate shape, then the vertices $V_{s+1}, \ldots, V_{r-1}$ lie outside the ϵ-envelope. In this case, we calculate the contribution of the distance of each such vertex V_k in the cost of the polygon. This is accomplished by calculating the distance $d_{vertex}^{V_k}$ of each vertex V_k with respect to the intermediate chain of edges $S_{i+1}, \ldots, S_{j-1}$ of the query shape augmented by the two previous and the two next edges:

$$s = S_{(i-2) \bmod m}, S_{(i-1) \bmod m}, ..., S_{(j+2) \bmod m}$$

This is consistent with the restriction of (ii). We also calculate the maximum distance $d_{edge}^{V_sV_r}$ of any point of the edge V_sV_r from the chain of edges s (see Figure 3).

(iv) We repeat these steps until we traverse all the vertices of the candidate shape P. At the end, we calculate the measure h_{vertex}^{P} as the average $d_{vertex}^{V_k}$, which is an approximation of the average distance of a vertex of the candidate shape P from the query shape. We also calculate the average $d_{edge}^{V_sV_r}$ over all the edges of the candidate shape that have both their edges inside the ϵ-envelope, which we denote by h_{edge}^{P}. The cost of the candidate shape P is defined as

$$h_P = \begin{cases} h_{vertex}^{P} & \text{if } h_{edge}^{P} \leq \epsilon \\ \max(h_{edge}^{P}, h_{vertex}^{P}) & \text{otherwise.} \end{cases}$$

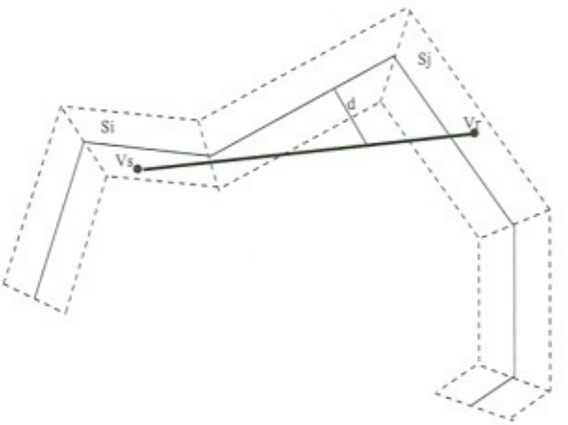


Figure 3: Calculating the maximum distance of the edge points from the three edges of the query shape

So finally we either have a valid candidate shape P with a cost h_P, or we have excluded this shape from the list of candidate shapes.

The worst case time complexity for this step is $O(\max(m, \kappa))$, where κ is the number of vertices of the candidate shape. In practice, $m \leq \kappa$ and therefore the time complexity is $O(\kappa)$. Then, the time complexity for doing the same traversal on all candidate shapes is $O(K_\epsilon)$, where K_ϵ is the total number of points within the ϵ-envelope.

3.4 Computing the initial width ($2\epsilon_s$) of the ϵ-envelope

Let l, p, and A be the length of the perimeter of the query shape, the number of shapes in the shape base, and the *active area*, i.e., the area in which the vertices of the normalized shapes fall. Since we use the diameter and we store copies for each shape normalized for pairs of points whose distance is at least α $(0 < \alpha \leq 1)$ times the length of the diameter, the area A is equal to the area of the lune defined by two circles of radius $(1 + \alpha)$ whose centers are at distance 1 apart. This implies that $A = (\frac{2\pi}{3} - \frac{\sqrt{3}}{2})(1 + \alpha)^2$.

By assuming uniform distribution of the vertices inside this lune, the average number of vertices inside an ϵ-envelope can be estimated to:

$$\bar{K}_\epsilon = \frac{2l\epsilon}{A}n$$

In order that the initial ϵ_s-envelope contain at least one candidate shape, we derive that:

$$\frac{\bar{K}_{\epsilon_s}}{(1-\beta)\frac{n}{p}} \geq 1 \Rightarrow \epsilon_s \geq \frac{(1-\beta)A}{2lp}$$

Since this is a conservative lower bound (we have the necessary number of vertices, but the probability that all of them belong to a single shape is very small) we determine experimentally a $\gamma(n)$, and we have that

$$\epsilon_s = \frac{(1-\beta)rA}{2ln}\gamma(n).$$

3.5 Increasing ϵ when there are no candidate shapes

In this case, all the shapes of our shape base have more than β of their vertices outside the current ϵ-envelope. Let us assume that the i-th shape P_i has at least q_i of its vertices outside the ϵ-envelope, where $q_i > \beta$. If s_i is the cost of P_i, then

$$s_i \geq \frac{t_i}{size(P_i)} + q_i$$

where t_i is the sum of the costs of the vertices that lie inside the envelope. For some new $\epsilon' > \epsilon$, the shape P_i will have $q_i' \leq q_i$ of its vertices outside the ϵ'-envelope and its cost s_i' will be

$$s_i' \geq \frac{t_i}{size(P_i)} + \frac{t_i'}{size(P_i)}$$

where t_i' is the cost of the $(q_i - q_i')$ vertices of P_i that are outside the ϵ-envelope but inside the ϵ'-envelope. Then we also have:

$$\frac{t_i'}{size(P_i)} + q_i\epsilon \leq \frac{t_i}{size(P_i)} + \frac{t_i'}{size(P_i)} + q_i'\epsilon'$$

$$\Rightarrow q_i'\epsilon \leq \frac{t_i'}{size(P_i)} + q_i'\epsilon'.$$

If we want $q_i' = \beta$ and since $(q_i - q_i')\epsilon \leq t_i' \leq (q_i - q_i')\epsilon'$,

$$\epsilon' \geq \frac{q_i}{\beta + \frac{q_i - \beta}{size(P_i)}}\epsilon.$$

Therefore, we increase ϵ according to this formula; if the number of vertices of the shapes in our shape base which fall inside the ϵ'-envelope does not increase sufficiently, we increase ϵ geometrically.

3.6 Geometric Hashing

When we cannot find a match and the envelope is large enough, suggesting that the matching algorithm has become inefficient, we revert to the *geometric hashing* method outlined in this section.

This method requires a finite family of curves which cover uniformly the *non-augmented active area* that contains the polylines of our shape base. The non-augmented active area is the intersection of two circles with unary radius, and is derived by considering that the diameter of every shape of the database has been normalized and rotated to fit $((0,0),(1,0))$. The augmented active area (or simply active area) contains some shape vertices which are not contained in the non-augmented active area; such vertices are considered to lie on the boundary of the non-augmented area for the purpose of this Section.

The non-augmented active area consists of four quarters Q_1, Q_2, Q_3 and Q_4 and is illustrated in Figure 4 (left). The uniform coverage of each quarter is achieved by requiring equal areas between consecutive curves of the family. The method works by assigning to each polygon/polyline of our shape base the closest curve segment of the family of curves. By increasing the number of curves, we are able to have a small, on the average, number of shapes assigned to each curve segment. Shapes that are close to each other will be assigned to neighboring curves. However, neighboring curves are not always assigned to similar shapes.

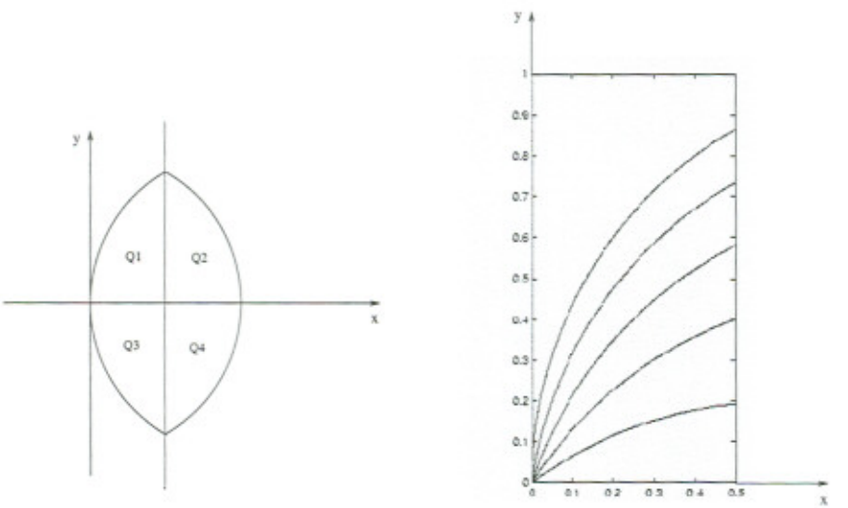


Figure 4: (left) the four quarters of the active area (right) hash curves for the upper left quarter

To retrieve similar matches for a given query shape, we calculate the curve

segment which is closest to this shape. Then, we retrieve all the shapes that have the same or neighboring hash curve. Finally, we apply the similarity measure between every retrieved shape and our query shape and report the shapes that exhibit the lowest costs.

We have considered different families of conic curves, trying to increase the retrieval accuracy, while minimizing the computational complexity of finding the curve closest to a given polyline. Thus, we have selected circles of constant unary radius that pass through the origin (since all shapes contain the origin) for the upper left quarter and likewise for the other three quarters. Finding the closest curve takes logarithmic time in the number of curves in the family. Figure 4 (right) shows some curves of this family for the upper left quarter.

We have generalized this method to make use of information from all four quarters. This is accomplished by assigning four curves to each shape, one for every quarter. Then, we consider shapes that have the same hash curve in any of the four quarters.

4 An Image Retrieval System

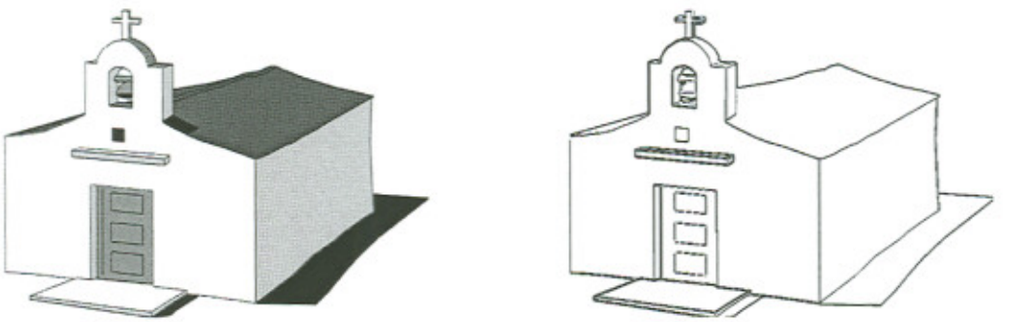
In this section we describe how the methods outlined in the previous section are integrated in an interactive image retrieval system.

4.1 Edge Extraction and Preprocessing

When adding a new image in the image base we process the image and extract shapes that describe sufficiently the boundary of each object. These shapes are non-selfintersecting polylines either open or closed. We first perform image processing that achieves edge extraction and enhancement. We then scan the image horizontally detecting clusters of polylines that describe the boundary of objects. Each such cluster consists of one or more non-selfintersecting polylines that share edges or vertices. Each time a cluster is detected, it is deleted from the processed image and the horizontal scanning continues.

In Figure 5 (a) we see an image and in 5 (b) we see the same image after it has been processed it for edge extraction. The result of the cluster detection is shown in Figure 6. There are seven clusters of shapes (A through G). The shapes are approximated by adequately small line segments, connected in polylines. Several heuristics may be used to minimize noise. Our method has been designed to be tolerant to such noise situations. Thus, noise elimination is important only for reducing the effective size of the shape base.

After the polyline clusters have been determined, each cluster is decomposed in a number of non-selfintersecting polylines. There are several different decompositions; achieving a good decomposition is an important issue, but we are not treating it here. During cluster detection and decomposition, certain relations



(a) (b)

Figure 5: (a) the original image (converted in grey scale) and (b) after the edge extraction

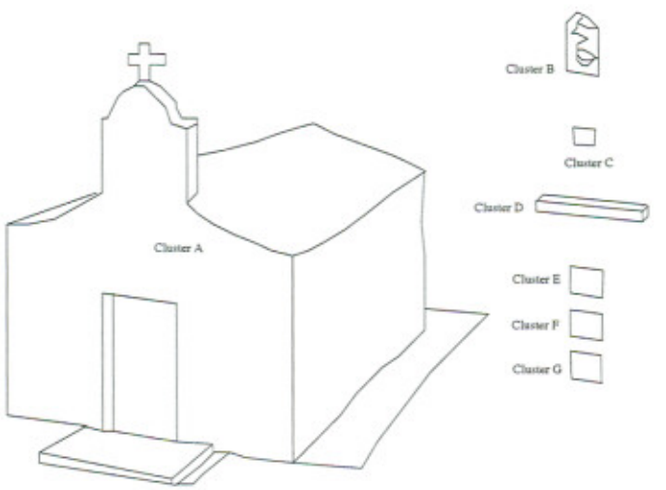


Figure 6: Seven clusters of shapes (A through G) have been detected

are recorded concerning the relative size and positioning of these shapes with respect to other shapes of the same or different clusters. The efficient use of this information (see e.g. [15])) is an important direction for future research.

4.2 User Dialogue

The overall user dialogue is illustrated in Figure 7. The user is first presented with a workspace where she/he can draft a query sketch. This sketch is then decomposed in non-selfintersecting polylines. Initially, the system attempts to use the incremental "fattening" algorithm to find the best match(es). If it fails to find a close match, geometric hashing is used for approximate retrieval. If the user is not satisfied by the returned result(s), she/he can edit the query sketch, specify certain polylines (open or closed) that are of special interest and re-apply the retrieval process.

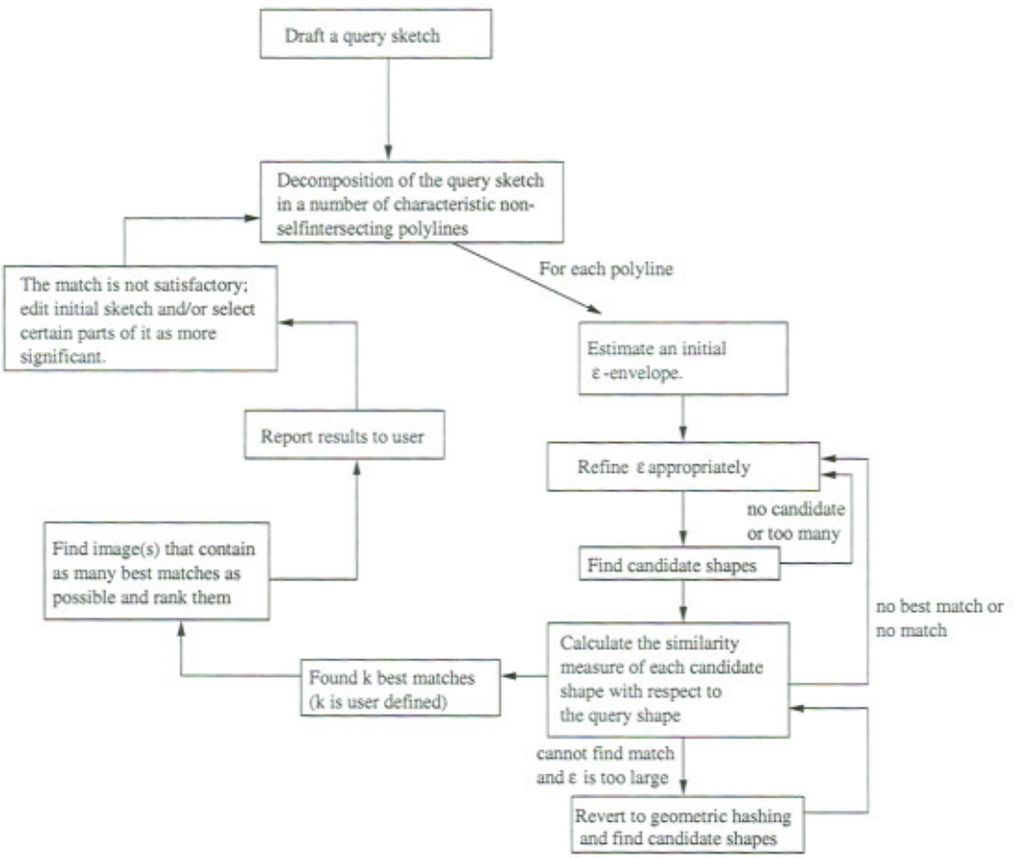


Figure 7: User Interaction

4.3 Implementation Issues

An efficient version of the algorithm is being developed in C. The user interface was developed using Tcl/Tk. We currently have a stable version running on Sun Solaris platform. We have performed experiments with around 100 images and 350 shapes. The software is easily portable to various platforms. Figure 8 depicts a snapshot of the GUI.

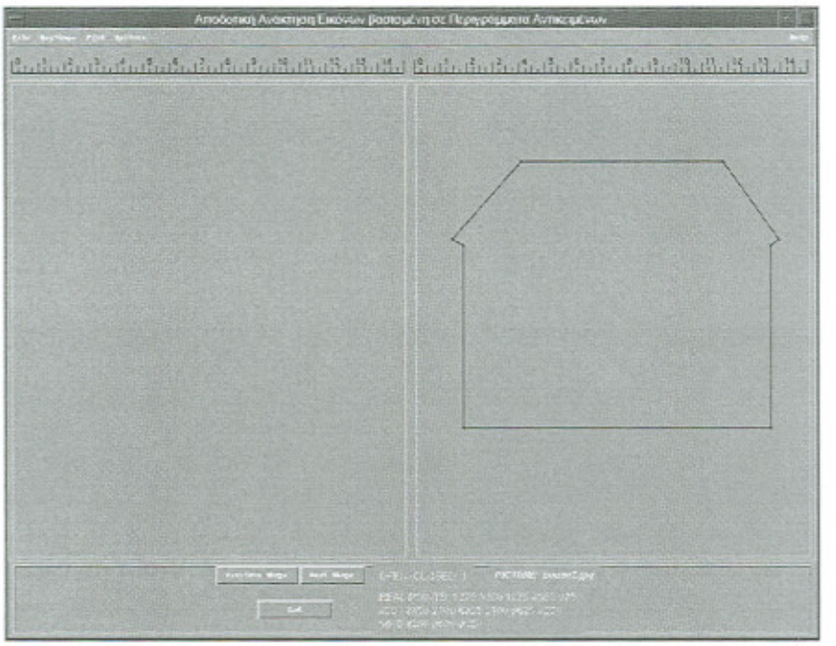


Figure 8: A snapshot of the GUI

5 Conclusions and Future Work

We have presented an efficient noise tolerant shape-based approach to image retrieval. Our system combines two powerful methods, an efficient algorithm for retrieving shapes based on a novel similarity criterion and a geometric hashing technique, to maximize efficiency and intuitiveness.

We are currently incorporating this approach in a video retrieval system. Future research directions include adding 3D awareness support, and using relative position information for allowing more complicated queries such as containment and tangency.

References

[1] M. Ankerst, H.P. Kriegel, and T. Seidl. Multistep approach for shape similarity search in image databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):996–1004, 1998.

[2] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kadem, and J.S.B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Knowledge and Data Engineering*, 13(3):209–216, 1997.

[3] A. Del Bimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):121–132, 1997.

[4] C. Carson and V. E. Ogle. Storage and retrieval of feature data for a very large online image collection. *IEEE Bulletin of the Tech. Comm. on Data Engineering*, 19(4):19–27, 1996.

[5] B. Chazelle and L.J. Guibas. Fractional cascading: I. a data structuring technique; ii. applications. *Algorithmica*, 1:133–191, 1986.

[6] S. Cohen and L. Guibas. Shape-based image retrieval using geometric hashing. In *Proceedings of the ARPA Image Understanding Workshop*, pages 669–674, May 1997.

[7] Ronald Fagin and Larry Stockmeyer. Relaxing the triangle inequality in pattern matching. *International Journal of Computer Vision*, to appear, 1999.

[8] M. Flincker, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. QBIC: Query by image and video content. *IEEE Computer*, 28(9):23–32, 1995.

[9] J. E. Gary and R. Mehrotra. Similar shape retrieval using a structural feature index. *Information Systems*, 18(7):527–537, 1993.

[10] J. E. Gary and R. Mehrotra. Feature-index-based similar shape retrieval. In S. Spaccapietra and R. Jain, editors, *Visual Database Systems*, volume 3, pages 46–65, 1995.

[11] J. E. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 1997.

[12] D. P. Huttenlocher and W. J. Rucklidge. A multi-resolution technique for comparing images using the hausdorff distance. Technical Report TR92-1321, CS Department, Cornell University, 1992.

[13] IBM. Ibm's query by image content (QBIC) homepage. http://wwwqbic.almaden.ibm.com.

[14] R. Mehrotra and J. E. Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, 28(9):57–62, 1995.

[15] M. Nabil, A.H.H. Ngu, and J. Shepherd. Picture similarity retrieval using the 2D projection interval representation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):533–539, 1996.

[16] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glassman, D. Petkovic, and P. Yanker. The QBIC project: querying images by content using color, texture and shape. In *Proc. SPIE Conference on Storage Retrieval for Image and Video Databases*, volume 1908, pages 173–181. SPIE, 1993.