Shape Retrieval Based on Dynamic Programming[*]

Evangelos Milios[†]
Dept. of Computer Science
York University
Toronto Canada
e-mail: `eem@cs.yorku.ca`

Euripides G.M. Petrakis[‡]
Dept. of Electr. and Comp. Engineering
Technical University of Crete
Chania, Crete, Greece
e-mail: `petrakis@ced.tuc.gr`

January 31, 2001

Abstract

We propose a shape matching algorithm for deformed shapes based on dynamic programming. Our algorithm is capable of grouping together segments at finer scales in order to come up with appropriate correspondences with segments at coarser scales. We illustrate the effectiveness of our algorithm in retrieval of shapes by content on two different two-dimensional datasets, one of static hand gesture shapes and another of marine life shapes. We also demonstrate the superiority of our approach over traditional approaches to shape matching and retrieval, such as Fourier descriptors, Geometric and Sequential moments. Our evaluation is based on human relevance judgements following a well established methodology from the information retrieval field.

Index Terms: image database, shape retrieval, query by example, dynamic programming, relevance judgements.

1 Introduction

Object recognition is an important problem in computer vision and has received considerable attention in the literature. Most approaches to object recognition are *model-based* [1], emphasizing the accuracy of recognition. They are limited to specific image types and require that all shapes are preprocessed and labeled prior to storage. However, the increasing amounts of image data in many application domains has generated additional interest for real-time management and retrieval of shapes [2, 3]. There, the emphasis is not only on accuracy, but also on efficiency (i.e., speed) of retrieval. Due to large image numbers, less emphasis may be given to preprocessing and labeling.

[*]IEEE Trans. on Image Processing, No. 1, Vol. 1, pp. 141-147, Jan. 2000.

[†]Corresponding author. Current address is Faculty of Computer Science, Dalhousie University, `eem@cs.dal.ca` This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

[‡]This work was carried-out while the author was visiting York University.

The performance of any object recognition system ultimately depends on the types of shape representations used and on the matching algorithms applied [4]. An important class of methods relies on symbolic entities obtained from the representation of the shape's inflection points at various scales and is referred to as *Curvature Scale Space (CSS)* representation [5]. In [6], matching is performed through *"interval trees"* which are computed by tracking the *CSS* representation from coarser to finer scales. In [7], matching is based on the maxima of *CSS* curves. Recently, matching in scale-space has been addressed with Dynamic Programming (DP) [8].

Regarding image database retrieval by shape content, [2] reports experiments with traditional shape representation and matching methods (i.e., Fourier descriptors, moment-based methods, combinations of such methods) on 500 trademark images. More recently, the effectiveness of such shape methods in conjunction with color features is investigated in [3] using 1,100 trademark images taken from two different datasets. In this work, we focus on shape content and shape similarity retrievals.

The contributions of this work are the following:

- We propose a shape matching algorithm which is particularly effective for shape retrieval.
- We establish the superiority of our method over traditional shape matching methods such as Fourier descriptors, Sequential and Geometric moments. We tested our algorithm on a data set of 980 two-dimensional hand gesture shapes and on a marine life database with 1,100 shapes.
- We introduce to the Computer Vision community a well established method from information retrieval for the empirical evaluation of retrieval results obtained by many competing methods.

We assume that shapes have already been extracted from images in the form of closed sequences of points. Automatic shape extraction from images (for example via region segmentation or edge following) is a non-trivial problem, and it is outside the scope of this paper. For our hand gesture dataset, contours are extracted from images by taking the polygonal approximation of the hand boundary after thresholding. The shapes of the marine dataset are already available in the desired form.

The major problem with segmented representations is that small perturbations to the shape can yield large changes in the segmentation. Therefore, the matching algorithm must be robust to segmentation changes. A standard fix is to represent the shape at multiple scales of resolution (smoothing), and either use a full scale-space representation for matching [6], or like [8] and our approach, have the algorithm choose the appropriate scales for different parts of the shape.

The rest of this work is organized as follows: Work related to our proposed algorithm is discussed in Section 2. Our shape matching algorithm is presented in Section 3. The database set-up, the evaluation criteria along with the experimental results are presented in Section 4. Conclusions and issues for future research are discussed in Section 5.

2 Related work

Our algorithm is a substantial extension of the DP algorithm of [8], in the following ways:

- We propose that the computation of the *CSS* representation be removed from the matching algorithm. The *CSS* representation has two drawbacks: First, it tends to diffuse the effects of a feature far away from its location as coarser scales are considered. This may be undesirable if such features have perceptual significance. Second, it is computationally expensive. Our algorithm is not only faster but also achieves matching accuracy comparable to that of the original formulation [9].

- Merging of neighboring segments in [8] is allowed only if such segments merge at some scale (not necessarily the same for the two shapes) in the *CSS* representation. We present a formulation of the algorithm that does not use scale space to restrict search for segment merges. Our algorithm allows all segment merges, and relies only on the minimization of the overall matching cost to select the merges.

- We have implemented a different set of cost measures from the original algorithm and have demonstrated their improved performance [9].

- The algorithm in [8] performs a best-only search in a DP table as it looks for minimum-cost paths in the DP framework. In [10] we have identified instances, in which a best-only search strategy fails to find a valid match between two shapes, although one exists [9]. We have extended the algorithm to perform k-best search and we have demonstrated that for a small k (e.g., 5), the additional space and time requirements are modest and the algorithm can solve matching problems where the original one fails (see also Section 3.5).

3 Shape matching

The shape matching algorithm that lies at the core of our methodology takes in two shapes and computes (a) their distance and (b) the correspondences between similar parts of the two shapes. In retrievals, only the distances are used. However, the correspondences help assess the plausibility of the distance computation, if necessary.

3.1 Definitions

Let A and B be the two shapes to be matched and let $A = a_0, a_1, \ldots a_{N-1}$ and $B = b_0, b_1, \ldots b_{M-1}$ be the convex/concave segment sequences of the two shapes, with a_i being the segment between two consecutive inflection points (i.e., points of change of curvature) p_i and p_{i+1}; similarly for b_j. Then, $a(i-n|i)$, $n \geq 0$, is the sequence of segments $a_{i-n}, a_{i-n+1}, ..., a_i$; similarly for $b(j-m|j)$, $m \geq 0$.

The algorithm searches for segment correspondences at various levels of shape detail by allowing matching of merged sequences of consecutive segments, if this leads to the minimization of a cost function. Each merging is a recursive application of the grammar rules $CVC \rightarrow C$ and $VCV \rightarrow V$, where C and V denote convex and concave segments respectively [11]. The number of merged segments is always odd. A *complete match* is a correspondence between groups of consecutive segments in order, such that no segments are left unassociated.

The goal is to find the *best* association of segments in shape A to segments in shape B. This is formulated as a minimization problem which is solved efficiently by dynamic programming: A table of partial costs is built and the optimal matching is searched in the form of a path in the DP table that minimizes a total dissimilarity cost.

The DP table has $2N$ columns and $2M$ rows, corresponding to segments of shape A and shape B respectively repeated twice (to force the algorithm to consider all possible relative rotations between the two shapes). All subscripts below are modulo N and M respectively. A link between cells (i_{w-1}, j_{w-1}) and (i_w, j_w) denotes the matching of segments $a(i_{w-1}+1|i_w)$ with $b(j_{w-1}+1|j_w)$. A *path* is a linked sequence of cells $(i_0, j_0), (i_1, j_1), ..., (i_w, j_w)$, not necessarily adjacent, indicating a partial match. A complete match has $(i_0, j_0) = (i_w, j_w)$. Function $\psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right)$ represents the dissimilarity cost between its two arguments and is defined in Section 3.3.

Each cell $cell(i, j)$ in the DP table contains the cost array $g[k_w]$ and associated bookkeeping data $t_1[k_w]$, $t_2[k_w]$, $index[k_w]$, $g_n[k_w]$, $g_m[k_w]$, where k_w varies from 0 to $K-1$ and refers to the k_w-th best path (k_w-th option), or partial match, up to and including $cell(i_w, j_w)$. Each cell keeps up to K best paths. Specifically, $g[k_w]$ holds the cost of the path, $t_1[k_w]$ and $t_2[k_w]$ hold the number of unmatched segments in shapes A and B respectively for the path and $index[k_w]$, $g_n[k_w]$ and $g_m[k_w]$ hold the back links for the path and allow the backward tracing of the path.

If $g_n[k_w] = n_w$ and $g_m[k_w] = m_w$ for $cell(i_w, j_w)$, then the previous cell and option in the path is $cell(i_{w-1}, j_{w-1}) = cell(i_w - 2n_w - 1, j_w - 2m_w - 1)$ and $index[k_w]$ respectively, where n_w and m_w are nonnegative integers. Notice that, $i_{w-1} = i_w - 2n_w - 1$ and $j_{w-1} = j_w - 2m_w - 1$, since the number of segments which are merged is always odd.

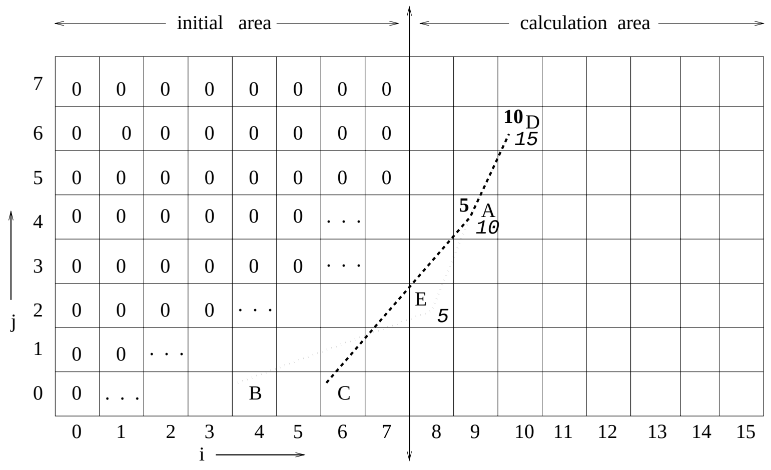

Figure 1: Example of a DP table.

Figure 1 illustrates the lower half of a DP table computed for the matching of two shapes with 8 segments each. Two incomplete paths are shown ending at $cell(10, 6)$. Numbers in cells indicate accumulated costs. The DP table consists of the initial value area (left half) and the calculation area (right half). In the initial value area all g terms are initialized to zero, t_1 to N and t_2 to M, implying that each of these cells can act as the first cell in a path. The calculation area is computed and finally the optimal path is searched. A path is complete when its corresponding t_1 and t_2 at the final cell of the path both become zero simultaneously.

3.2 The algorithm

The main idea in the algorithm is to fill the DP table cells by scanning forward and then search for the optimal complete path by tracing backward. We outline the algorithm below.

```
for $i_w = N, N+1, \ldots 2N-1$ do
    for $j_w = 0, 1, \ldots 2M-1$ do
        fill the K options in $cell(i_w, j_w)$;
        check if a complete path has been found;
    end for
end for
find the complete path with the lowest cost;
retrace segment matches by following the backward links;
```

The *for* loop for j_w does not run over all the indicated values, but only over those values that do not involve convex to concave segment associations, which are not possible. To describe the filling of the k_w-th option cost $g[k_w]$ of $cell(i_w, j_w)$ of the DP table, we need to compute the following $f(n, m, k)$ for all possible n, m and k, and select the k_w-th smallest value.

$$\begin{aligned} f(n,m,k) = g(i_w - 2n - 1, j_w - 2m - 1, k) + \\ \psi\left(a(i_w - 2n|i_w), b(j_w - 2m|j_w)\right), \quad n, m \geq 0, \quad 0 \leq k < K. \end{aligned} \tag{1}$$

The cost ψ in the above equation is the cost of association of segments $a(i_w - 2n|i_w)$ with $b(j_w - 2m|j_w)$ and consists of a merging cost component and a dissimilarity cost component (see Section 3.3 below). Additional constraints on acceptable pairs (n, m) are that either $t_1 = t_2 = 0$ (a complete match has been found), or $t_1 > 0, t_2 > 0$ (there exist unmatched segments in both shapes).

3.3 Cost components

The cost term ψ in Equation 1 can be rewritten as $\psi(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w))$. Following the notation of [8], this cost term consists of three additive components:

$$\begin{aligned} \psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right) = \\ MergingCost(a(i_{w-1}+1|i_w)) + \\ MergingCost(b(j_{w-1}+1|j_w)) + \\ \lambda\; DissimCost\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right), \end{aligned} \tag{2}$$

where λ represents the relative importance of the merging and dissimilarity costs. In this work λ was set to 1.

The first two terms in Equation 2 represent the cost of merging segments $a(i_{w-1}+1|i_w)$ in shape A and segments $b(j_{w-1}+1|j_w)$ in shape B respectively while, the last term is the cost of associating the merged $a(i_{w-1}+1|i_w)$ with the merged $b(j_{w-1}+1|j_w)$.

Requirements for reliable cost computation are the following:

- Merging should follow the process grammar rules [11] (i.e., each allowable merging should be a recursive application of the grammar rules $CVC \Rightarrow C$ and $VCV \Rightarrow V$). This is enforced by the DP algorithm.

- Merging a *"visually prominent"* segment (i.e., a large segment with high curvature) into a merged segment of the opposite type (convex or concave) should incur a high cost. To specify this requirement, we need to define *visual prominence* in geometric terms.

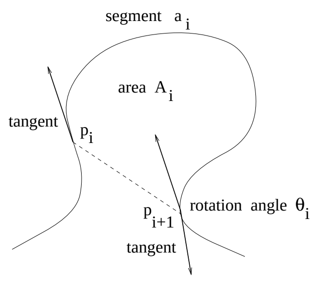

Figure 2: Geometric quantities for defining the prominence of a segment

- The partial cost components arising from different features of the shape should be combined into a total cost in a meaningful way.

The heuristic cost computations that follow attempt to satisfy the above requirements. Firt, we define geometric quantities (features) needed in the specification of visual prominence of a segment according to Figure 2.

Rotation Angle θ_i is the angle traversed by the tangent to the segment from inflection point p_i to inflection point p_{i+1} and shows how strongly a segment is curved.

Length l_i is the length of segment a_i.

Area a_i is the area enclosed between the chord and the arc between the inflection points p_i and p_{i+1}.

Dissimilarity cost computation: We assign a higher cost to segments (or groups of segments) with large differences in more than one feature:

$$DissimCost = W \max_{all\ features\ f} \{d_f\}, \tag{3}$$

where $f = l, \theta$ or a. We choose the max operation instead of product (as in [8]) because in the product a small cost in terms of one feature can cancel the effect of a high cost in terms of another feature, something that may lead to a visually implausible outcome. The max operation addresses this problem. Factor W equals the number of features for which d_f is greater than $0.75 \times \max\{d_f\}$, where $f = l, \theta, a$. Thus, if all three features have uniformly large d_f, then the dissimilarity cost is multiplied by 3.

The term d_f, for $f = \theta$ is defined as

$$d_f = \left| \frac{\Theta_A - \Theta_B}{\Theta_A + \Theta_B} \right|, \tag{4}$$

where $\Theta_A = \sum_{s=i_{w-1}+1}^{i_w} \theta_s$, and $\Theta_B = \sum_{s=j_{w-1}+1}^{j_w} \theta_s$, and θ_s being the rotation angle of segment with index s of shape A and shape B respectively.

The term d_f, for f being l (length) or area (a), is defined as

$$d_f = \left| \frac{f_A}{F_A} - \frac{f_B}{F_B} \right|, \tag{5}$$

where $F_A = \sum_{s=0}^{N-1} f_s$, $f_A = \sum_{s=i_{w-1}+1}^{i_w} f_s$ of shape A and similarly for F_B, f_B of shape B.

Merging cost computation: Let the types of the segments being merged be $CVC\dots VC$. The opposite case is obtained by switching C and V. The merging cost is defined as follows:

$$MergingCost = \max_{all\ features\ f} \{w_f c_f\}, \tag{6}$$

where subscript f refers to a feature (length, area or rotation angle). We choose the above maximization formula instead of sum of products of terms comparing consecutive segments (as in [8]) for the following reasons: We used max instead of product, because in a product a small cost in terms of one feature can cancel the effect of a high cost in terms of another. The reason for abandoning the sum of consecutive segments is because it implies that the plausibility of merging several segments can be reduced to the similarity of consecutive segments, which may not necessarily be true. Consider, for example, the case of a short and flat segment next to a long and curved one. In this case, it is plausible to merge the two, while the merging cost in [8] will be high. Another drawback of the use of a sum is that the merging cost increases with the number of segments merged, even if several very short segments are being merged into a large one.

For f being length or area:

$$c_f = 1 - \frac{\sum_{C\ segs\ of\ group} f - \sum_{V\ segs\ of\ group} f}{\sum_{all\ segs\ of\ shape} f} \tag{7}$$

For f being rotation angle:

$$c_f = 1 - \frac{\sum_{C\ segs\ of\ group} f - \sum_{V\ segs\ of\ group} f}{\sum_{C\ segs\ of\ group} f + \sum_{V\ segs\ of\ group} f} \tag{8}$$

The intuition behind these formulae is that they measure the visual prominence of the features of the absorbed segments (of type V) relative to the absorbing segments (of type C). All costs c_f are within the interval $[0, 2]$. Cost c_f is close to 0 if the convex segments visually dominate the concave ones (hence it is plausible to absorb the concave ones), while it is close to 2 if the concave segments visually dominate the convex ones (hence it is not plausible to perform the merge, therefore the merging cost should be high).

For f being any feature (length, area, rotation angle) the weight term is

$$w_f = \frac{N}{2} \frac{\sum_{V\ segs\ of\ group} f}{\sum_{V\ segs\ of\ shape} f}. \tag{9}$$

The intuition behind the weight term is to measure the visual prominence of the absorbed segments within the shape as a whole. Factor $\frac{N}{2}$ is heuristic.

3.4 Matching examples

Figure 3 illustrates segment correspondences (indicated by consecutive lines connecting the starting and ending points of the associated segments) obtained by matching hand silhouettes (left) and fish silhouettes (right). One of the shapes has been scaled to 50% of the original. Notice that the algorithm tried to come-up with plausible segment associations by matching groups of segments which, in the case of the fish shapes are due to shape detail.

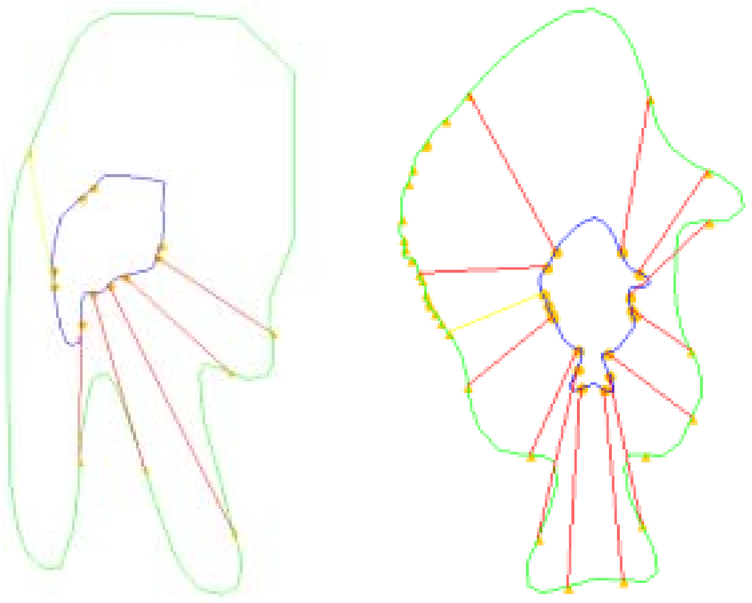

Figure 3: Segment associations reported by the matching algorithm on representative matches from the gestures and the marine life databases.

3.5 K-option DP search

The algorithm in [8], except that it uses the *CSS* representation to constrain the search for possible matches, is a special case of the above K-option DP algorithm for $K = 1$. The motivation for introducing the K-option method is that, by using only one option at each cell, the optimal path is sometimes missed. This happens because the (n, m) pair chosen at $cell(i, j)$ is the one which leads to the least cost path only up to $cell(i, j)$. This path may not be the one that leads to the formation of a complete path. It is possible that after the entire DP table is computed, there exist only incomplete paths, in which case the algorithm fails to produce a match. Also, the best choice at $cell(i, j)$ may lead to an expensive match of the remaining unmatched segments. This is illustrated in figure 1, which shows the DP table for matching two shapes with 8 and 4 segments respectively. Cells in the table are marked as follows: $(4, 0)$ as B, $(6, 0)$ as C, $(8, 2)$ as E, $(9, 4)$ as A and $(10, 6)$ as D. At each cell, the number in bold represents the cost of the path up to that cell using the $K = 1$ option (dashed line). Numbers in italics represent the cost along the alternate path ($K = 2$, dotted line). Alternate path $BEAD$ is ignored by the $K = 1$ option algorithm because of a least cost choice (i.e. $CEAD$) made at A, although this may lead to an incomplete path. This problem is addressed by the K-option DP method where the choice of a best subpath is not made at cell A but deferred until a later point when more information is available.

4 Shape retrieval

In our experiments we used the following datasets:

- GESTURES [1]: Consists of 980 synthetic shapes which are generated from 17 original hand gestures. We took the 17 shapes in pairs and, for each such pair, we produced a number of blended shapes by transforming one shape to the other using the shape morphing algorithm of [12]. To evaluate our method we took the 17 original shapes as queries.

- SQUID [2]: Consists of 1,100 shapes of marine species. We carefully selected 20 shapes from the SQUID database and we used them as queries.

Each shape is represented by its contour. All contours are preprocessed to contain between 80-100 points. The fish silhouettes of the SQUID database contain much finer contour detail than the hand silhouettes of the GESTURES database.

The experiments are designed to illustrate the superiority of our shape matching algorithm over traditional methods of shape matching and retrieval. All measurements below correspond to averages over 17 and 20 queries respectively.

4.1 Comparison with other methods

The competitors to our method are:

Fourier descriptors [13] : This is known to be one of the most successful methods for the recognition of closed shapes. We computed the first (lower order) 20 coefficients of the Fourier transform.

Sequential moments [14]: This is one of the most effective moment-based methods for closed shapes. For each shape, a representation of 4 moment coefficients is computed from its bounding contour.

Geometric moments [15]: This is the original and the most characteristic representative of a wide class of methods based on area moments. A representation of 7 moment coefficients of the shape is computed from the area it occupies.

Additional reasons for choosing these methods for our evaluation are: (a) They are translation, rotation and scale invariant (the same as our method) and (b) They all filter out shape detail so that, they can detect shape similarity at a coarse view-scale. Our method has these advantages too but, in addition, matching with our method which is *local* (as opposed to *global* matching with the above methods) reporting all associations between similar segments choosing (possibly) different scales for different parts of the shapes depending on noise or shape detail.

Fourier descriptors, Sequential and Geometric moments are pre-computed and stored in separate files in the database along with the original contours. When a query is given, its representation is computed and it is matched with similar representations stored in the database. This, typically takes less than less than 5 seconds on a SUN Ultra 1 for each data set. For our method, no pre-computed information is stored. Instead the actual shape contours are used to search the database. For this reason, our method is the slowest requiring approximately 1

[1]We have made our database available on the Internet at `http://www.cs.yorku.ca/~eem-/gesturesDB`.

[2]SQUID is available from `http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/de-mo.html`.

second per shape match on a Pentium PC, 200MHz. Certain optimizations that could speed-up our method are possible, such as the precomputation and storage of the convex and concave segments of all shapes in the database, and the selection of a number of options K that ensures optimal tradeoff between accuracy and amount of computation.

4.2 Evaluation criteria

We used human relevance judgements to compute the effectiveness of each method. Two shapes (i.e., a query and a stored shape) are considered similar if a human judges that they represent the same figure. To measure effectiveness, for each candidate method we computed:

Precision , defined as the percentage of similar shapes retrieved with respect to the total number of retrieved shapes.

Recall , defined as the percentage of similar shapes retrieved with respect to the total number of similar shapes in the database. Because we don't have the resources to compare every query with each database shape (i.e., this would require, for each method, $17 \times 980 = 16,660$ visual judgements for the GESTURES dataset and $20 \times 1,100 = 22,000$ visual judgements for the SQUID dataset!), for each query, we merged the answers obtained by all candidate methods and we considered this as the database which is manually inspected for relevant entries. Notice however, that this method does not allow for absolute judgements such as *"method A misses 10% of the total similar answers in the database"*. It provides, however, a fair basis for comparisons between methods allowing judgements such as *"method A returns 5% fewer correct answers than method B"*.

4.3 Experimental results

Each query retrieves the best 50 shapes. For answer sets containing between 1 and 50 entries, we computed the average values of precision and recall. Precision and recall values are represented in *precision-recall plots*: The horizontal axis corresponds to recall and the vertical axis corresponds to precision. Each method is represented by a curve. Each point in such a curve is the average over 17 queries (for the GESTURES database) and 20 queries (for the SQUID database) respectively. The total number of points in each curve is 50 (i.e., we compute precision and recall for answers containing between 1 and 50 shapes). Therefore, the top-left point of the diagram corresponds to the precision/recall values for the best answer (best match), while the bottom right point corresponds to the precision/recall values for the entire answer set with 50 retrieved shapes.

Figure 4 illustrates the precision-recall diagram for the GESTURES database. For small answer sets returning up to 12 shapes (corresponding to the left-most 12 points of a curve) our method and Fourier descriptors perform about equally in terms of both, precision and recall. Notice that, for such small answer sets, both methods achieve precision close to 1, that is, their answers are almost 100% correct. For larger answer sets, our method performs clearly better than any other method, achieving up to 25% better recall and 20% better precision than the second best method (Fourier descriptors). This result demonstrates that our method is very well suited for image retrieval where one is (typically) interested in retrieving more than 10 or 20 and up to 50 shapes.

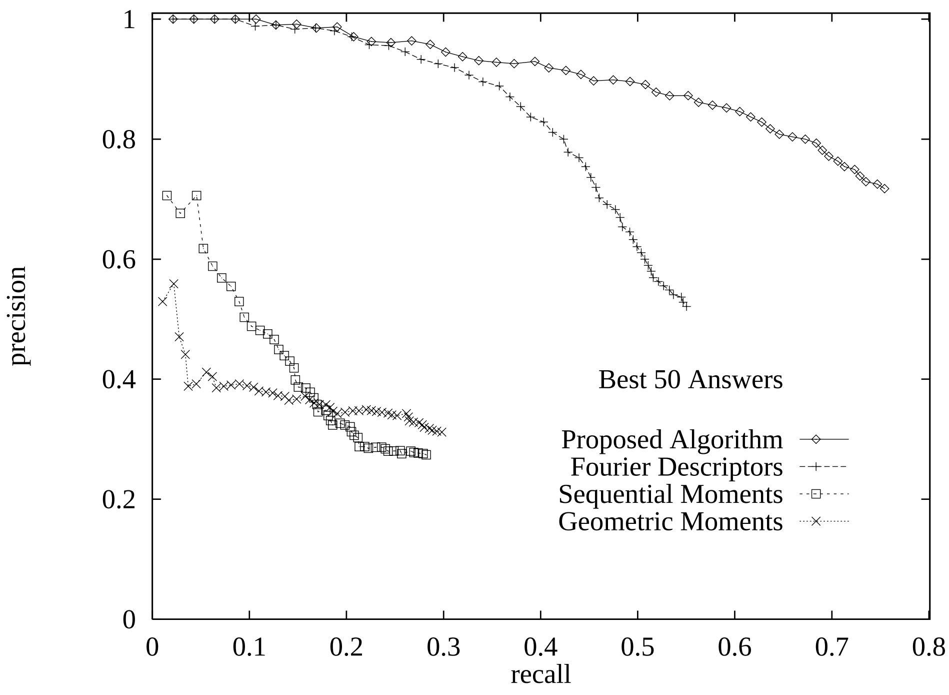

Figure 4: Precision-Recall diagram for the GESTURES database corresponding to (a) Our shape matching algorithm, (b) Fourier descriptors, (c) Sequential moments and (d) Geometric moments.

Figure 4 illustrates the precision-recall diagram for the SQUID database. Our method performs clearly better than any other method, achieving up to 18% better recall and 15% better precision. An important observation is that all methods achieved lower values of precision and recall than those achieved on the GESTURES database. Presumably, all methods are sensitive to noise and contour detail which mainly exist in the shapes of the SQUID database.

We see that our method outperforms the others both the GESTURES and the SQUID databases. Although it is slower than its competitors, the selection of a method for shape retrieval should be based mainly on effectiveness (i.e., quality of results) rather than on efficiency (i.e., speed).

5 Conclusions

We propose a shape matching algorithm for handling shape similarity retrievals in image databases. Our algorithm is based on dynamic programming, performs implicitly at multiple scales and allows the matching of deformed shapes.

We demonstrate the superiority of our approach over traditional approaches to shape matching and retrieval (Fourier descriptors, Geometric and Sequential moments) using two different datasets with 980 and 1,100 shapes respectively. We also introduce to the Computer Vision community a well-established methodology for the evaluation of the retrieval results obtained by more than one competing methods.

Current research is directed towards extending our matching algorithm for open curves. Future work includes the experimentation with more datasets and methods, and the handling of combined queries involving more than one feature (e.g., shape, color, text). Recently,

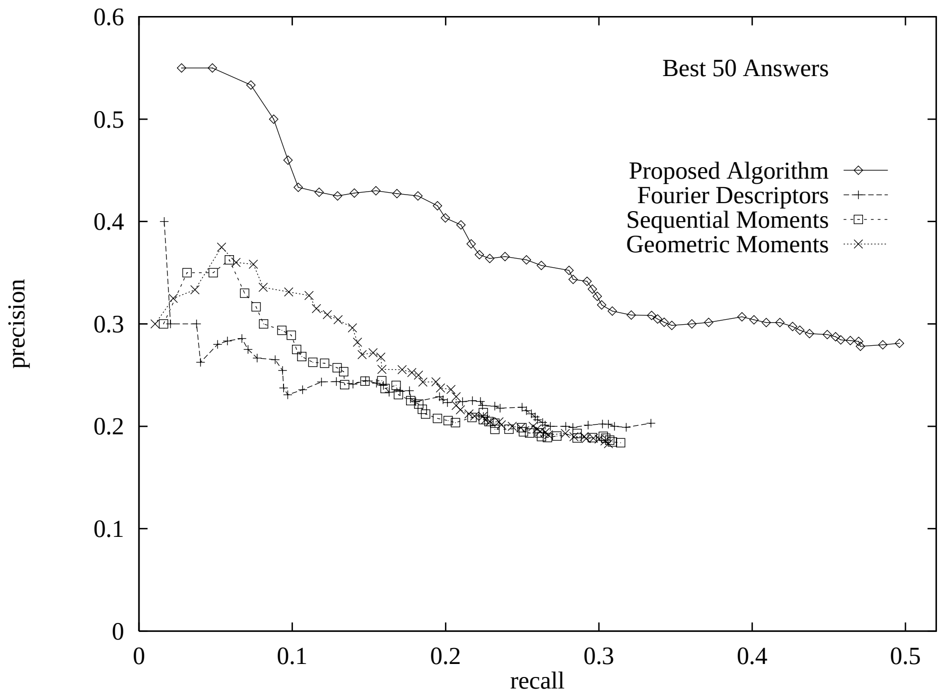

Figure 5: Precision-Recall diagram for the SQUID database corresponding to (a) Our shape matching algorithm, (b) Fourier descriptors, (c) Sequential moments and (d) Geometric moments.

we have proposed an indexing mechanism for our method which achieves up to three orders of magnitude speed-up over sequential scanning with very high accuracy, providing also for clustering visualization and browsing of a data set [9].

Acknowledgements

We are grateful to Pantelis Elinas, for his help in the experiments, Zusheng Rao for valuable discussions and for a shape morphing program to generate the GESTURES database. Marita Roussou and Panos Economopoulos implemented the shape matching algorithm in C++. Jyoti Baid introduced the K-option idea. We are also grateful to Prof. Mokhtarian of the Centre for Vision, Speech and Signal Processing laboratory at the University of Surrey, UK, for providing the SQUID database for our experiments.

References

[1] Roland T. Chin and Charles R. Dyer. Model-Based Recognition in Robot Vision. *ACM Computing Surveys*, 18(1):67–108, 1986.

[2] Babu M. Mehtre, Mohan S. Kankanhalli, and Wing Foon Lee. Shape Measures for Content Based Image Retrieval: A Comparison. *Information Processing and Management*, 33(3):319–337, 1997.

[3] Anil K. Jain and Aditya Vailaya. Shape-Based Retrieval: A Case Study With Trademark Image Databases. *Pattern Recognition*, 31(9):1369–13990, 1998.

[4] Sven Loncaric. A Survey of Shape Analysis Techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[5] A. Witkin. Scale Space Filtering. In *Proceedings 8th IJCAI, (Karlsruhe, West Germany)*, pages 1019–1022, 1983.

[6] Farzin Mokhtarian and Alan Mackworth. Scale-Based Description of Plannar Curves and Two-Dimensional Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):34–43, 1986.

[7] F. Mokhtarian. Silhouette-Based Object Recognition through Curvature Scale Space. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(5), May 1995.

[8] Naonori Ueda and Satoshi Suzuki. Learning Visual Models from Shape Contours Using Multiscale Convex/Concave Structure Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):337–352, April 1993.

[9] Evangelos Milios and Euripides Petrakis. Efficient Matching and Retrieval at Multiple Scales. Technical Report CS-1998-11, Dept. of Comp. Science, December 1998. (http://www.cs.yorku.ca/techreports/1998/CS-1998-11.html).

[10] J. Baid and E. Milios. Deformed Shape Recognition using Dynamic Programming. Technical report, Department of Computer Science, York University, 1995.

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

[12] Thomas W. Sederberg and Eugene Greenwood. A Physical Approach to 2-D Shape Bending. *Computer Grphics*, 26(2):25–34, 1992.

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

[14] L. Gurta and M. D. Srinath. Contour Sequence Moments for the Classifiaction of Closed Planar Shapes. *PR*, 20(3):267–271, 1987.

[15] Ming-Kuei Hu. Visual Pattern Recognition by Moment Invariants. *IRE Transactions in Information Theory*, IT-8:179–187, 1962.