Efficient Retrieval of Deformed and Occluded Shapes

Zusheng Rao[1] Euripides G.M. Petrakis[2] Evangelos Milios[1] *
[1] Dept. of Computer Science
York University, Toronto Canada, M3J 1P3
[2] Dept. of Electronic and Comp. Engineering
Technical University of Crete, Chania, Crete, Greece
e-mail: zrao@cs.yorku.ca, petrakis@ced.tuc.gr, eem@cs.dal.ca

Abstract

We propose an approach for matching deformed and occluded shapes using Dynamic Programming (DP). Our algorithms handle noise and shape distortions by allowing matching of merged sequences of consecutive small segments in a shape, with larger segments of another shape, while being invariant to translation, scale and orientation transformations of shapes. We illustrate the effectiveness of our algorithms in retrieval of shapes on two different two-dimensional datasets, one of static hand gesture shapes and another of marine life shapes.

1. Introduction

The increasing amounts of image data in many application domains has generated additional interest for real-time management and image database retrieval by shape content. For example, [2] reports experiments with traditional shape representation and matching methods (e.g., Fourier, moments) on 500 trademark images. More recently, the effectiveness of such methods in conjunction with color features is investigated in [1] using 1,100 trademark images. SQUID [5] supports retrievals on a dataset of 1,100 marine life species. In [4] we demonstrate the superiority of a multi-scale DP matching method over Fourier and moment-based methods on 980 static hand gesture shapes and on the dataset of SQUID. A promising method, but untested in shape retrieval, defines a shape distance and association of shape parts on the basis of area features (shape skeleton) [7], by reducing shape matching to a largest subgraph isomorphism problem.

*This work was carried-out while E. Petrakis was visiting York University. Current address of E. Milios 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.

In this work, we propose shape matching algorithms for deformed and occluded shapes based on dynamic programming. The methods referred to above do not focus on occlusion and, as noted in [6], earlier DP shape matching methods (e.g., [8, 4]) are not optimal, that is, they may miss the optimal match. Our algorithms are optimal, in that they always find the least cost match [6]. We tested our algorithms on a dataset of 980 two-dimensional hand gesture shapes and on a marine life database with 1,500 shapes.

The rest of this work is organized as follows: The main idea behind our approach is given in Section 2. The DP table and the definition of the cost functions are discussed in Section 3. Our shape matching algorithm is presented in Section 4. Finally, experimental results are presented in Section 5 followed by conclusions and issues for future research in Section 6.

2. Main idea

The shape matching algorithm that lies at the core of our methodology takes in two shapes and computes: (a) Their distance; the more similar the shapes are, the lower the value of the distance function and (b) The correspondences between similar parts of the two shapes.

In matching two shapes A and B, the algorithm builts a DP table, where rows and columns correspond to segments of A and B respectively. Starting at the lower left corner and proceeding upwards and to the right, the table is filled with the cost of the partial match containing the segments (rows and columns) swept so far. Because convex segments cannot match concave ones, only about half the cells are assigned cost values, in a checkerboard pattern. Merges, where a segment sequence of one shape matches a single segment of the other shape can occur. Merges introduce "jumps" in the traversal of the DP table. Reaching the top row implies a complete match, where all segments of shape A has been swept. Additional information is stored in each cell to allow the tracing of a path starting from that cell and working backwards. The tracing of a path reveals segment associations between the two shapes.

2.1. Matching cases

We distinguish among the following three cases of matching:

Both shapes are open: The algorithm will find the *best* association of all segments of A to all or to a subsequence of segments of B (i.e., part of B may be left unmatched) or vice versa. Because we cannot know in advance which shape is included within the other one, we run the algorithm twice (i.e., once for each possibility) and we take the matching with the minimum cost.

Shape A is open and shape B is closed: The algorithm will find the best association of all segments of A to all or to a subsequence of segments of B. Shape A may be contained within shape B, but not the other way around; this is the only possibility (i.e., part of B may be left unmatched).

Both shapes are closed: The algorithm will find the *best* mapping between A and B so that, no segments remain unassociated in either shape.

We have addressed the third case in [4]. That algorithm is not optimal (i.e., may miss the least cost match). In this paper, we handle the case with A and B closed by pretending that A is open, repeating the algorithm for open and closed shape matching M times (once for each possible starting point on A) and by taking the least cost match as the cost of matching. In the following, we focus on the first two cases.

3. Definitions and methodology

Let A and B be the two shapes to be matched. Let $A = a_1, a_2, \ldots a_M$ and $B = b_1, b_2, \ldots b_N$ be the sequence of N and M convex (C) and concave (V) segments of the two shapes respectively, with a_i being the segment between inflection points p_i and p_{i+1} and b_j the segment between inflection points q_j and q_{j+1}. Henceforth, $a(i-m|i)$, $m \geq 0$, denotes the sequence of segments $a_{i-m}, a_{i-m+1}, ..., a_i$; similarly for $b(j-n|j)$, $n \geq 0$.

3.1. Dynamic Programming (DP) table

The DP table has $\mathcal{M}$ rows and $\mathcal{N}$ columns, where $\mathcal{M}$ and $\mathcal{N}$ are defined as follows:

Both shapes are open: $\mathcal{M} = M + 1$ and $\mathcal{N} = N + 1$.

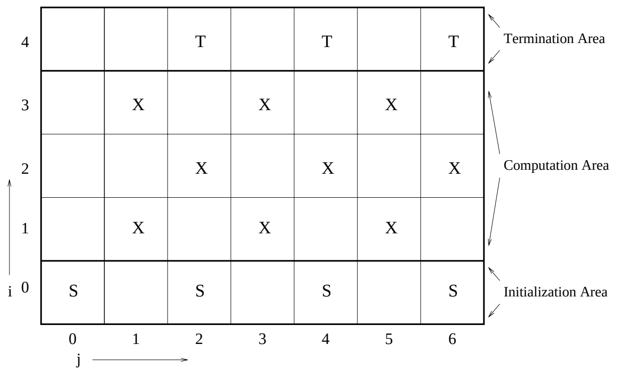

Figure 1. Example of a DP table with $\mathcal{M} = 4$ (shape A) and $\mathcal{N} = 6$ (shape B). S, X and T denote cells in the initialization, computation and termination areas respectively.

Shape A is open and shape B is closed: $\mathcal{M} = M+1$ and $\mathcal{N} = 2N$. Shape B is repeated twice to force the algorithm consider all possible starting points on B. If B is closed, its indices are taken modulo N. If A is closed and B is open, we switch the roles of A and B.

Both shapes are closed: This case reduces to the previous one.

The rows of a DP table are indexed by i, $0 \leq i \leq \mathcal{M}$ and its columns are indexed by j, $0 \leq j \leq \mathcal{N}$ where, i, j are indices to segments of A and B respectively. The first row ($i = 0$) and the first column ($j = 0$) do not correspond to segments of A and B respectively, but they are placeholders for initializing the filling of the DP table.

The cell at the intersection of row i and column j is referred to as $cell(i,j)$. A link between cells (i_{w-1}, j_{w-1}) and (i_w, j_w) denotes the matching of the merged sequence of segments $a(i_{w-1}+1|i_w)$ with $b(j_{w-1}+1|j_w)$. $cell(i_{w-1}, j_{w-1})$ is called *parent* of $cell(i_w, j_w)$. A *path* is a linked sequence of cells $\big((i_0, j_0), (i_1, j_1), \ldots (i_t, j_t)\big)$, not necessarily adjacent, indicating a partial match, where $i_0 < i_1 < \ldots < i_t$ and $j_0 < j_1 < \ldots < j_t$. This path associates segment sequences $a(i_{w-1}+1|i_w)$ of A with sequences $b(j_{w-1}+1|j_w)$ of B for $w = 1, 2, \ldots t$.

Each $cell(i_w, j_w)$ contains the following values: w, $g(i_w, j_w)$, m_w, n_w, u_w, v_w and ρ_w where, w is the number of matched sequences of segments, $g(i_w, j_w)$ is the partially accumulated match cost up to that cell, u_w and v_w denote number of unmatched segments of A and B respectively, m_w and n_w are the indices of the parent cell of $cell(i_w, j_w)$ (i.e., $m_w = i_{w-1}$ and $n_w = j_{w-1}$) and are used to trace back a path. Finally, ρ_w denotes the scale factor corresponding to the parts of A and B which have been matched up to $cell(i_w, j_w)$ and it is defined in Section 3.3.

Fig. 1 illustrates an example of a DP table. The DP table consists of three distinct areas:

Initialization area: It is first row of the DP table. If a_1 and b_j, $1 \leq j \leq \mathcal{M}$, have the same polarity, then $w, g(0,j), m_w, n_w, u_w, v_w, \rho_w$ of $cell(0,j)$ are $0, 0, 0, 0, M, N, 1$ respectively. If a_1 and b_j have different polarity, we set $g(0,j) = \infty$.

Computation area: It is the area after the first column and between the first and last row of the DP table. Cells in this area correspond to incomplete paths.

Termination area: It is the last row of the DP table. All complete paths end at cells in this area. The best match corresponds to the path with the least cost.

3.2. Cost of matching

A *complete match* is a correspondence between sequences of segments in order, such that no segments are left unassociated in shape A and there are no crossovers or omissions. A complete match is characterized by a *complete path* $((i_0, j_0), (i_1, j_1), ..., (i_W, j_W))$, a path that starts at the initialization and ends at the termination area. The cost $D(A,B)$ of matching shape A with shape B is defined as

$$D(A,B) = \min_{(i_w, j_w)} \sum_{w=1}^{W} \psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right). \tag{1}$$

Function $\psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right)$ represents the dissimilarity cost of its two arguments and consists of three additive components:

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

The first two terms in Eq. 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 sequence $a(i_{w-1}+1|i_w)$ with the merged sequence $b(j_{w-1}+1|j_w)$. The role of merging cost is to restrict non promising merges. Constant λ represents the relative importance of the merging and dissimilarity costs. In this work λ was set to 1. Each allowable merging should be a recursive application of the grammar rules $CVC \Rightarrow C$ and $VCV \Rightarrow V$) [3]. This is enforced by the DP algorithm.

Next, we define geometric quantities (features) required in the definition of the cost functions as illustrated in Fig. 2.

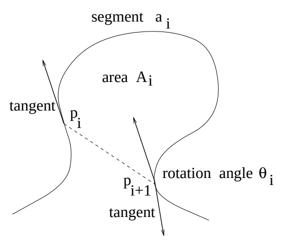

Figure 2. Geometric quantities for defining the importance of a segment

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}.

3.3. Scale factor

If one of the two shapes is scaled with respect to the other, then the length of one of the two shapes (i.e., shape B) has to be multiplied by an appropriate *scale factor*. Although we know that the matched part of shape A will be the whole shape A, the length of the matched part of shape B is unknown before the algorithm is completed. To handle this problem, we compute a scale factor ρ_t for each partial match $((i_0, j_0), (i_1, j_1), ..., (i_t, j_t))$:

$$\rho_t = \frac{\sum_{w=1}^{t} \sum_{i=i_{w-1}}^{i_w} l_i(A)}{\sum_{w=1}^{t} \sum_{j=j_{w-1}}^{j_w} l_j(B)}, \tag{3}$$

where $1 \leq t \leq W$ and $l_i(A)$ and $l_j(B)$ are the lengths of a_i and b_j respectively. This value is an approximation of the actual scale factor of a complete match. Notice that ρ_0 is undefined since the total matched length is 0 for both shapes. In this work ρ_0 is set to 1.

3.4. Dissimilarity cost

The cost of associating a group of segments from shape A with a group of segments from shape B is computed as

$$DissimCost = W \max_{all\ features\ f} \{d_f\}. \tag{4}$$

The term d_f is the cost associated with the difference in feature f (i.e., length, area or angle). The intuition behind the use of max is that it tends to emphasize large differences on any feature. W is a weight term associated with the importance of this partial match. Specifically, the proportion of the matched shape length with respect to the total length is used to define W:

$$W = \max\left\{ \frac{\sum_{i=i_{w-1}+1}^{i_w} l_i(A)}{length\ of\ A},\ \frac{\sum_{j=j_{w-1}+1}^{j_w} l_j(B)}{length\ of\ B} \right\}. \tag{5}$$

The term d_f is defined as

$$d_f = \frac{|F_A - S_w(f)F_B|}{F_A + S_w(f)F_B}, \tag{6}$$

where, $F_A = \sum_{i=i_{w-1}+1}^{i_w} |f_i|$, $F_B = \sum_{j=j_{w-1}+1}^{j_w} |f_j|$ and $S_w(f)$ is a parameter depending on the feature f. Specifically, $S_w(f) = \rho_{w-1}$ for f being length and ρ_{w-1}^2 for f being area. For f being rotation angle, $S_w(f) = 1$ since, angle measurements do not depend on the scale factor.

3.5. Merging cost

Let the types of the segments being merged be $CVC\ldots C$. In the following, the opposite case is obtained by switching C and V in the formulaes. The merging cost is defined as follows:

$$MergingCost = \max_{all\ features\ f} \{W_f C_f\}, \tag{7}$$

where subscript f refers to a feature (length, area or rotation angle). For all features:

$$C_f = \frac{\sum_{V\ segs\ of\ group} |f|}{\sum_{all\ segs\ of\ group} |f|}. \tag{8}$$

The intuition behind these formulaes is that they measure the importance of the absorbed segments (of type V) relative to the whole matched consecutive segments of the group. For f being any feature (length, area, rotation angle) the weight term of the merging cost is defined as

$$W_f = \frac{\sum_{V\ segs\ of\ group} |f|}{\sum_{V\ segs\ of\ shape} |f|}. \tag{9}$$

The intuition behind this weight term is to measure the importance of the absorbed segments within the shape as a whole.

4. Algorithm

Fig. 3 outlines the algorithm. The for loop for j_w does not run over all the indicated values, as convex to concave matches are not possible. In fact, only half of the cells are used. At each cell, the algorithm computes the optimum cost of the incomplete path ending at this cell:

$$\begin{aligned} g(i_w, j_w) = & \\ \min\{ \quad & g(i_{w-1}, j_{w-1}) + \\ & \psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right)\}. \end{aligned} \tag{10}$$

```
// Initialization: Fill the first row
for $j_w = 0, 1, \ldots, \mathcal{N}$ do
    fill $cell(0, j_w)$
end for
// Fill from the 2nd to the $\mathcal{M}$-th row
for $i_w = 1, 2, \ldots, \mathcal{M}$ do
    for $j_w = 1, 2, \ldots, \mathcal{N}$ do
        fill $cell(i_w, j_w)$ using $\rho_{w-1}$, Eq. 2 and Eq. 10;
        compute $\rho_w$ using Eq. 3;
    end for
end for
// Select the least cost complete path
    select the least cost path from the $\mathcal{M}$-th row;
    retrace path using $m_w$, $n_w$ cell values;
```

Figure 3. Outline of the algorithm.

Merging always involves an odd number of segments that is, $(i_{w-1}, j_{w-1}) = (i_w - 2m_w - 1, j_w - 2n_w - 1)$, where $m_w \geq 0$ and $n_w \geq 0$. Eq. 10 determines the minimum cost transition from cell $cell(i_{w-1}, j_{w-1})$ to $cell(i_w, j_w)$ for all possible values of m_w and n_w. Indices m_w and n_w are stored at $cell(i_w, j_w)$ and can be used to retrace the path from $cell(i_w, j_w)$ back to its starting point.

4.1. Matching examples

Fig. 4 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 two shapes has been shrunk and rotated properly to illustrate the associations between matched parts of the two shapes.

5. Shape retrieval

In our experiments we used the following datasets:

- GESTURES[1]: Consists of 980 open shapes which are generated from a dataset of 980 closed shapes by editing.
- MARINE[2]: Consists of 1,500 open shapes of marine species which we generated from the 1,100 closed shapes of SQUID[3].

To evaluate our algorithm we carefully created 34 queries (17 closed and 17 open shapes) for each dataset. We used

[1]http://www.cs.yorku.ca/~eem/gesturesDB.
[2]http://www.cs.yorku.ca/~eem/marineDB.
[3]http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html.

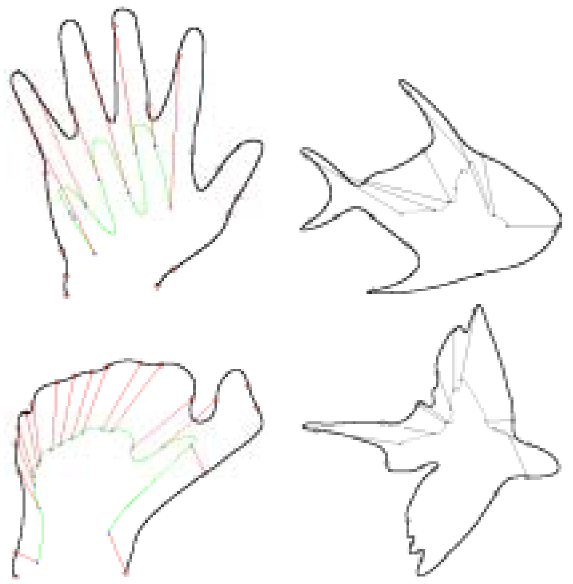

Figure 4. Segment associations reported by the matching algorithm on representative matches from the gestures and the marine life datasets.

human relevance judgements to compute the effectiveness of our method. Two shapes (i.e., a query and a stored shape) are considered similar if a human judges that they represent the same figure or that the one is contained within the other. To measure effectiveness, we computed *precision*, that is the percentage of similar shapes retrieved with respect to the number of retrieved shapes.

Retrieved shapes	1	5	10	15	20
GESTURES	0.91	0.80	0.75	0.70	0.65
MARINE	0.80	0.52	0.45	0.40	0.38

Table 1. Average values of precision for the gestures and the matine life datasets.

Table 1 illustrates the average values of precision for the above two datasets and for answers containing between 1 up to 20 shapes. This result demonstrates that our method is best suited for applications where one is interested in retrieving a few best matches. Notice that the algorithm achieves always at least 10% better precision on the GESTURES dataset than on the MARINE dataset. Presumably, this is because the shapes in the MARINE dataset have much more shape detail and noise and are much more difficult to handle than the shapes in the GESTURES dataset. The algorithm requires less than 1 second per shape match on the average on a Pentium PC, 200MHz.

6. 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 by allowing the matching of merged sequences of segments and handles noisy, deformed and occluded shapes. Future work includes the experimentation with more datasets and methods, the handling of combined queries involving more than one feature (e.g., shape, color, text), the development of indexing methods that could speed-up retrievals and the development of a graphical user interface on the World Wide Web.

Acknowledgements

We are grateful to Aristidis Diplaros for his help in the experiments and Prof. Mokhtarian of the Centre for Vision, Speech and Signal Processing laboratory at the University of Surrey, UK, for providing us the marine dataset.

References

[1] A. Jain and A. Vailaya. Shape-Based Retrieval: A Case Study With Trademark Image Databases. *Pattern Recognition*, 31(9):1369–13990, 1998.
[2] B. Mehtre, M. Kankanhalli, and W. Lee. Shape Measures for Content Based Image Retrieval: A Comparison. *Information Processing and Management*, 33(3):319–337, 1997.
[3] E. Milios. Shape Matching using Curvature Processes. *Computer Vision, Graphics and Image Processing*, 47:203–226, 1989.
[4] E. Milios and E. Petrakis. Shape Retrieval Based on Dynamic Programming. *IEEE Trans. on Image Processing*, 9(1):141–147, 2000.
[5] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and Robust Retrieval by Shape Content through Curvature Scale Space. In *Proceedings of Int. Workshop on Image DataBases and MultiMedia Search*, pages 35–42, Amsterdam, The Netherlands, 1996.
[6] Z. Rao, E. Milios, and E. Petrakis. Retrieval of Deformed and Occluded Shapes using Dynamic Programming. TR CS-1999-06, Department of Computer Science, York University, Toronto, 1999. http://www.cs.yorku.ca/techreports/1999/CS-1999-06.html.
[7] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock Graphs and Shape Matching. *Int. Journal of Computer Vision*, 35(1):13–32, 1999.
[8] N. Ueda and S. Suzuki. Learning Visual Models from Shape Contours Using Multiscale Convex/Concave Structure Matching. *IEEE PAMI*, 15(4):337–352, April 1993.