Retrieval of Deformed and Occluded Shapes using Dynamic Programming

Zusheng Rao
Euripides Petrakis
Evangelos Milios

Technical Report CS-1999-06

December 3, 1999

Department of Computer Science
4700 Keele Street North York, Ontario M3J 1P3 Canada

Retrieval of Deformed and Occluded Shapes using Dynamic Programming

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: petrakis@ced.tuc.gr, eem@cs.dal.ca *

December 3, 1999

Abstract

We propose an approach for matching deformed shapes using dynamic programming. 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. Our proposed algorithms handle occlusion while being invariant to translation, scale and orientation transformations of shapes. We

*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.

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. Our evaluations are based on human relevance judgements and the results are a good support to our claims of accuracy.

Index Terms: shape matching, occlusion, dynamic programming, shape retrieval, image database, query by example.

1 Introduction

Shape matching is a central problem in pattern recognition and computer vision research and has received considerable attention in the literature [1]. Most approaches to object recognition are *model-based* [2], 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 [3, 4]. There, the emphasis is not only on accuracy, but also on efficiency (i.e., speed) of retrieval. Little or no emphasis is given to preprocessing and labeling in this case.

A wide range of shape recognition approaches have been proposed, such as structural (e.g., methods organizing local features into graphs [5], trees [6] or strings [7]), fuzzy or probablistic (e.g., relaxation methods [8]), statistical (e.g., methods based on moments [9]), methods that work on some transform domain (e.g., Fourier [10] or Hough [11]), and methods based on Neural Networks [12, 13]. An important class of contour tracking and matching methods relies on physical models of the deformation and is based on minimization of an energy function, without first extracting a symbolic representation of the shapes [14, 15, 16, 17, 18].

Multiscale methods are considered the most promising, due to their ability to take into account variations in shapes due to different degrees of smoothness. Various forms

of scale space descriptions have been proposed [19, 20, 21]. In an earlier approach [22], matching is performed through *"interval trees"* which are computed by tracking the *"Curvature Scale Space" (CSS)* representation [19] from coarser to finer scales. In [23] only the maxima of the *CSS* curves are used. In [24] it is demonstrated, however, that small shape changes may cause major structural changes in the interval tree and this may lead to matching errors. Recently, multiscale methods have been combined with Dynamic Programming (DP) [24, 25].

Regarding image database retrieval by shape content, experiments have been reported with traditional shape representation and matching methods (i.e., Fourier descriptors, moment-based methods, combinations of such methods) on 500 trademark images [3] . More recently, the effectiveness of such shape methods in conjunction with color features is investigated in [4] using 1,100 trademark images taken from two different datasets. In [25] we demonstrate the superiority of a multiscale DP matching method over Fourier and moment-based methods, on two different datasets, one of 980 static hand gesture shapes and another of 1,100 marine life species. However, none of these methods handle occluded objects.

In this work we focus on DP programming methods and we propose shape matching algorithms for occluded and deformed shapes. If the two shapes are scaled with respect to each other, the algorithm determines the appropriate scale for matching. The algorithms find the *best* association between segments of one shape and segments of the other shape. This is formulated as a minimization problem which is solved efficiently by dynamic programming: A table of partial costs is built and the optimal complete matching is searched in the form of a path in the DP table that minimizes a total dissimilarity cost. Our algorithms are optimal, that is, they always find the least cost path. We report experimental results of our algorithms on a data set of 980 two-dimensional hand gesture shapes and on a marine life database with 1,500 shapes.

We assume that shape contours have been already extracted from images. Automatic shape contour extraction from images (e.g., 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 polygonal 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 [22], or have the algorithm choose the appropriate scales for different parts of the shape [24]. Our approach is to favor the merging of small segments into larger segments which are more likely to match single segments of the appropriate size in the less noisy shape.

The rest of this report is organized as follows: In Section 2 we review work related to Dynamic Programming for shape matching and to the shape representations used. Background information, such as basic notations and definition of cost functions, is given in Section 3. Our shape matching algorithm is presented in Section 4. The database set-up, the evaluation criteria along with the experimental results are presented in Section 5. Conclusions and issues for future research are discussed in Section 6.

2 Related Work

In the following, we review work on dynamic programming for shape matching and the underlying representations.

Recently, matching of contours, together with detection of contours in image sequences, has been addressed with dynamic programming In [26, 27] dynamic programming is used to minimize a cost function that accounts for displacement of a contour in a pair of images from an image sequence. In [28, 18], dynamic programming is used to fit a closed curve template to an image (deformable template matching).

Another class of matching methods relies on symbolic entities extracted from shape

contours [29, 30, 23]. Dynamic programming has been a popular approach for matching such symbolic entities [31, 32, 33, 34]. In [31] the inability of dynamic programming to combine contour segments is mentioned, as well as the fact that differing resolutions in the matched contours will lead to reduced performance. In [33] deletions and insertions of features (corners in a polygonal representation) as well as smoothing of features (i.e. dropping corners) is incorporated in the dynamic programming scheme. This type of smoothing lends a primitive multiple-scale character to the method. In [32] dynamic programming is used to guide the application of grammar rules that transform one shape into another, in the spirit of [35]. In [34], matching proceeds both forward and backward from a support match between two features (landmark points) that are maximally similar. Features are extracted based on their persistence across scales. However, there is no matching of features at multiple scales. Therefore, multiple scales are used as a preprocessing stage only.

Building upon the previously mentioned work, [24] is a sophisticated dynamic programming algorithm which can group segments together in order to come up with appropriate correspondences. This algorithm uses the scale space representation of [19] to constrain the possible merges (i.e., it accepts merges that are only present at coarser scales of the scale space representation). The algorithm in [25] is a substantial extension of the above algorithm to perform k-best search as it searches for best matches in the DP framework while avoiding the expensive computation of zero crossings in scale space. Both of these algorithms work only for closed shapes and, as noted in [25], neither algorithm is optimal, that is, they may miss the optimal match.

3 Background

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 curves are, the lower the value of the distance function and (b) The correspondences between similar parts of the two shapes. In retrievals, only distances between shapes are used. However, the

correspondences help assess the plausibility of the distance computation, if necessary.

3.1 Shape preprocessing

A shape is represented by discrete sample points computed from its surrounding contour. We get a smooth B-spline list approximation of each such point list and we compute the inflection points (i.e., points of change of curvature) on this spline representation. If the shape is open, there is no guarantee that there will be two inflection points close to its two end points. If not, we get a representation of the part of the original shape which is enclosed between its first and last inflection points. To handle this problem we produce additional inflection points around the two end points by adding small *sin* (or *cos*) curves. The scale of *sin* curves is very small compared to shape length. At most one extra inflection point is kept. Figure 1 shows an example of this pre-processing. The *sin* curves have been enlarged for viewing.

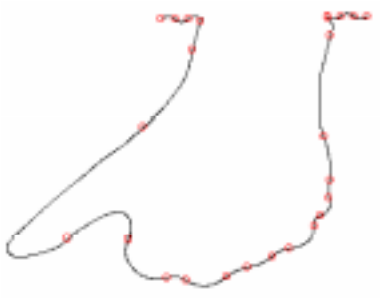

Figure 1: *Preprocessing of the shape's end points.*

3.2 Basic notations

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 $b(j-n|j)$, $n \geq 0$.

If shape A (or shape B) is closed, then $p_1 = p_{M+1}$ (or $q_1 = q_{N+1}$). This implies that the number of inflection points in closed shapes equals the number of segments. If shape A (or shape B) is open, then $p_1 \neq p_{M+1}$ (or $q_1 \neq q_{N+1}$). This implies that the number of inflection points of open shapes equals the number of segments plus 1.

3.3 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 a subsequence of segments of B (i.e., part of shape 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 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 (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 [25]. 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: shape A is open and it is included within B (which can be either open or closed).

3.4 The Dynamic Programming (DP) table

The algorithm builts a DP table of costs of partial matches and a match between A and B is searched in the form of a path in the DP table that minimizes a total dissimilarity cost. The DP table has $\mathcal{M}$ rows and $\mathcal{N}$ columns, where $\mathcal{M}$ and $\mathcal{N}$ are defined based on the number of segments of the two shapes as follows:

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

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.

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

The rows of a DP table are indexed by i, $1 \leq i \leq \mathcal{M}$ and its columns are indexed by j, $1 \leq j \leq \mathcal{N}$ where, i, j are indices to inflection points of A and B respectively. If shape B is closed, its indices are taken modulo N.

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 $b(j_{w-1} + 1|j_w)$. A *path* is a linked sequence of cells $((i_0, j_0), (i_1, j_1), ..., (i_t, j_t))$, not necessarily adjacent, indicating a partial match, where $i_0 < i_1 < ... < i_t$ and $j_0 < j_1 < ... < j_t$. This path begins at inflection point p_{i_0} of shape A and at inflection point q_{j_0} of shape B and tries to match sequences of segments $a(i_{w-1}+1|i_w)$ of A with sequences $b(j_{w-1}+1|j_w)$ of B for $w = 1, 2, ..., t$. The previous cell of $cell(i_w, j_w)$ is denoted by $cell(i_{w-1}, j_{w-1})$ and is called *parent* of $cell(i_w, j_w)$.

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. Fi-

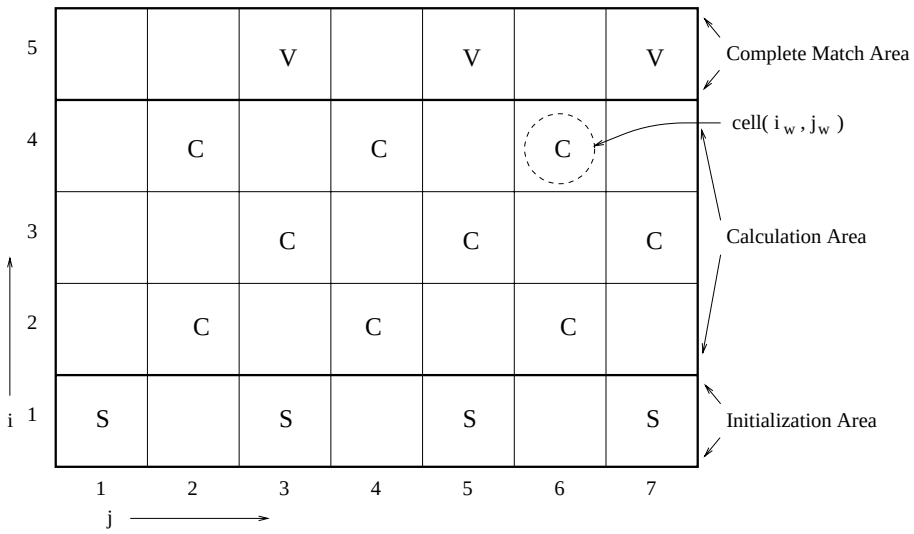

Figure 2: *Example of a DP table with $\mathcal{M} = 6$ (shape A) and $\mathcal{N} = 8$ (shape B). S, C and V denote cells in the initialization, computation and complete match areas respectively.*

nally, 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 3.5.

Figure 2 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. Cost terms in this area are initialized appropriately. All paths start from cells in this area.

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

Complete match 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.

Notice that about half of the cells of the above DP table are empty; this is because associations between opposite type segments (i.e., C and V) are not allowed. By convention, the cost of matching C with V segments is infinite. Notice also that the first column is empty except the cell in the first row. The reason for this is that we

do not allow empty merge. Finally, we assumed that both, the first segment of A and the first segment of B have the same polarity; otherwise matching will start from the second segment of B.

3.5 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. 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 complete match 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:

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

The first two terms in 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 sequence $a(i_{w-1}+1|i_w)$ with the merged sequence $b(j_{w-1}+1|j_w)$. Constant λ represents the relative importance of the merging and dissimilarity costs. In this work λ was set to 1. From the definition of cost components below follows that the total cost of any complete path is within the range of $[0, 2+\lambda]$.

Requirements for reliable cost computation are the following:

- Merging should follow the process grammar rules [32] (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.

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

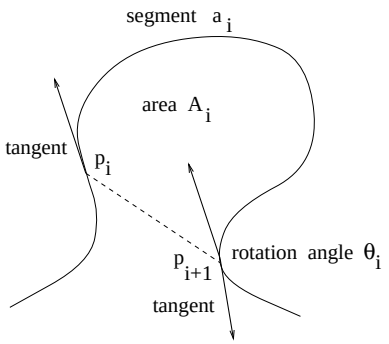

Figure 3: Geometric quantities for defining the prominence of a segment

We define geometric quantities (features) needed in the specification of visual prominence of a segment according to Figure 3.

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}.

3.6 Scale factor

If one of the two shapes is scaled with respect to the other, the length of one of the two shapes (i.e., shape B in this work) has to be multiplied by an appropriate *scale factor*. This scale factor can be computed as the ratio of the lengths of the matched parts of shapes A and B respectively. We distinguish between the following two cases:

Global matching: Shape A matches the whole shape B. The algorithm consumes all segments from both shapes. Referring to Figure 2, the cost of matching will be always that of the rightmost cell of the complete match area. The scale factor is constant and is computed as

$$\rho = \frac{length\ of\ A}{length\ of\ B}. \tag{3}$$

Equivalently, we can normalize initially both shapes with respect to their perimeter. It is easy to accommodate this to our method by omitting all scale terms ρ from the algorithm.

Local matching: Shape A may match either the whole or only a part of shape B. This case is more difficult to handle but, it is more general and includes the previous one when matching the whole shape B yields the least cost. 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 path $((i_0, j_0), (i_1, j_1), ..., (i_t, j_t))$, corresponding to the matched parts:

$$\rho_t = \frac{\sum_{w=1}^{t} \sum_{i=i_{w-1}}^{i_w} l_i(A)}{\sum_{w=1}^{t} \sum_{j=j_{w-1}}^{j_w} l_j(B)}, \tag{4}$$

where $0 < t \leq W$ 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.

Henceforth, we refer to the second (more general) case, where shape A is open and B is closed.

3.7 Dissimilarity cost

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

$$DissimilarityCost \;=\; W \max_{all\ features\ f}\{|d_f|\}. \tag{5}$$

The term d_f is the cost associated with the difference in feature f (i.e., length, area or angle). Notice that, for f being angle, d_f can be negative. 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 dissimilarity of this partial match: W will emphasize the importance of matching large parts from both shapes similarly to the way humans pay more attention on large shape parts when judging the quality of matching. Without W, the matching of very small shape parts contribute to the cost of matching equally to the matching of very large parts. Specifically, the proportion of the matched shape length with respect to total length is used to define W:

$$W = \max\left\{\frac{\sum_{i=i_{w-1}+1}^{i_w} l_i(A)}{length\ of\ A},\ \ \frac{\sum_{j=j_{w-1}+1}^{j_w} l_j(B)}{length\ of\ B}\right\}. \tag{6}$$

Area can also be used instead of the length for the definition of W. However, rotation angle should not be used since a large (small) portion of a shape might have a small (large) rotation angle.

The term d_f is defined as

$$d_f = \frac{|F_a - S_w(f)F_b|}{F_a + S_w(f)F_b}, \tag{7}$$

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. ρ_{w-1} is computed according to Equation 4 or Equation 3 for local and global matching respectively. When matching is local and for $w = 1$, we put $d_f = 0$ since ρ_0 can not be defined (i.e., there is no information to estimate the scale factor). For f being rotation angle, $S_w(f) = 1$ since angle measurements do not depend on the scale.

3.8 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 formulas. The merging cost is defined as follows:

$$MergingCost = \max_{all\ features\ f} \{W_f C_f\}, \tag{8}$$

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

For all features:

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

The intuition behind these formulaes is that they measure the visual prominence of the features of the absorbed segments (of type V) relative to the whole matched consecutive segments of the group. All costs C_f are within the interval $[0, 1]$.

For f being any feature (length, area, rotation angle) the weight term of the merging cost is defined as

$$W_f = \frac{\sum_{V\ segs\ of\ group} |f|}{\sum_{V\ segs\ of\ shape} |f|}. \tag{10}$$

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

4 The algorithm

In the following, first we outline our algorithm and then we discuss issues related to optimality and complexity. Finally, we give examples of matching.

4.1 Outline

Figure 4 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:

$$g(i_w, j_w) = \min_{m_w, n_w} \left\{ g(i_{w-1}, j_{w-1}) + \psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right)\right\}. \quad (11)$$

Merging always involves an odd number of segments that is $(i_{w-1}, j_{w-1}) = (i_w - 2m_w - 1, j_w - 2n_w - 1)$. Then, Equation 11 can be rewritten as

$$g(i_w, j_w) = \min_{m_w, n_w} \left\{ g(i_w - 2m_w - 1, j_w - 2n_w - 1) + \psi\left(a(i_w - 2m_w|i_w), b(j_w - 2n_w|j_w)\right)\right\}, \quad (12)$$

where $0 \leq m_w \leq \frac{i_w - 1}{2}$ and $0 \leq n_w \leq \frac{j_w - 1}{2}$. Equation 12 determines the minimum cost transition from cell $cell(i_{w-1}, j_{w-1})$ to $cell(i_w, j_w)$. Indices (m_w, n_w) characterize this transition. They 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.

Matching always starts at the first inflection point of A ($i_0 = 1$) while any point of B is a candidate starting point. We initialize the DP table by filling its first row: When a_1 and b_j have the same polarity, then $g(1, j), m_1, n_1, u_1, v_1$ are $0, 0, 0, M, N$ respectively, implying that each of these cells can act as a starting point. If a_1 and b_j have different polarity, we set $g(1, j)$ to be ∞ since the matching of opposite type segments (C or V) is not allowed (no path begins at these cells). If B is closed, then for $N < j \leq \mathcal{N}$, we set $g(1, j) = \infty$ since cell indices j are modulo N ($b_j = b_{j-N}$) and we don't have to calculate the same starting point twice.

Notice that $u_w > 0$ and $v_w > 0$ implies that the match has not consumed neither shape A nor shape B; $u_w = 0$ and $v_w \geq 0$ implies that the match has consumed all segments of shape A and some (or all) segments of shape B that is, a complete match has been found. Notice that, $u_w \geq 0$ and $v_w \geq 0$ implies $i_w \leq \mathcal{M}$ and $j_w \leq \mathcal{N}$, which is automatically satisfied when the $cell(i_w, j_w)$ is in the DP table. It is easy to see that $u_w = 0$ or $v_w = 0$ implies that $i_w = \mathcal{M}$ or $j_w = \mathcal{N}$ respectively.

// **Initialization: Fill the first row**

for $j_w = 1, 2, ..., \mathcal{N}$ **do**

    fill $cell(1, j_w)$ using Equation (2);

    calculate ρ_1 using equation (4);

end for

// **DP Propagation: Fill from the second to the $\mathcal{M}$-th row**

for $i_w = 2, 3, ..., \mathcal{M}$ **do**

    for $j_w = 1, 2, ..., \mathcal{N}$ **do**

        fill $cell(i_w, j_w)$ using ρ_{w-1} and Equations (2), (11);

        calculate ρ_w using Equation (4);

    end for

end for

// **Select the least cost complete path**

    select the least cost complete path from the $\mathcal{M}$-th row;

    retrace path using m_w, n_w cell values;

Figure 4: *Outline of the algorithm.*

4.2 Optimality

Now we prove that our shape matching algorithm is optimal, that is, it always finds the path with the least cost. The proof makes use of the DP table of Figure 5 showing two alternative complete match paths.

Lemma 1 *The algorithm of Section 4 is optimal.*

Proof: Let $P = ((i_0, j_0)(i_1, j_1)...(i_W, j_W))$ be the best (least cost) complete path that the algorithm has found. If this is not the least cost path then, let this be path $P' = ((i'_0, j'_0), ...(i_W, j_W))$. This path ends at (i_W, j_W) too, since we always select (i_W, j_W) as the end point of the path with the least cost. The two paths meet at some

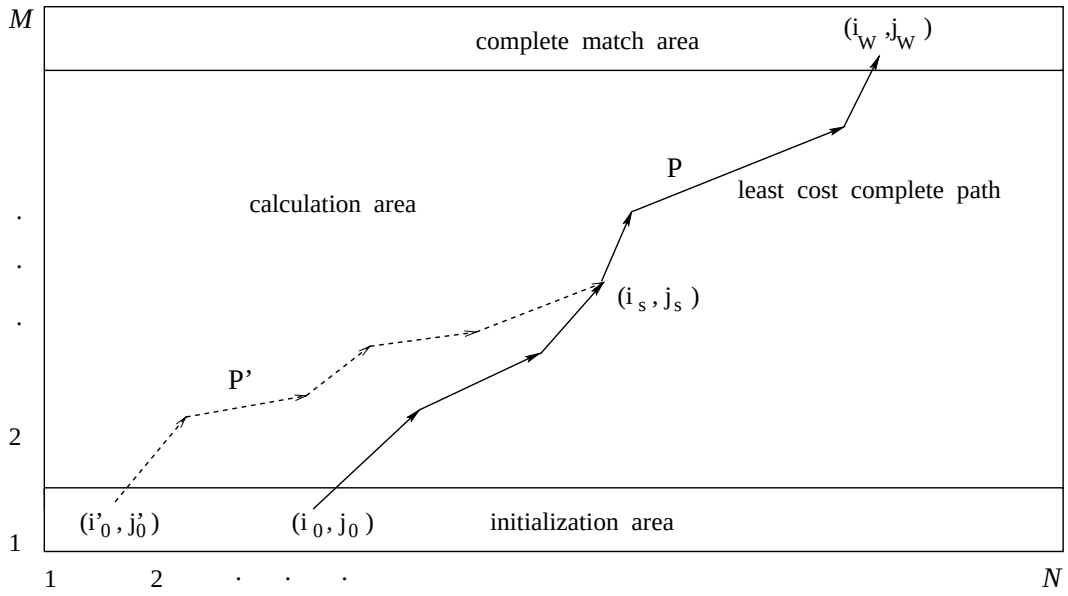

Figure 5: *Example of DP table showing two alternative complete match paths.*

point (i_s, j_s) either before or at (i_W, j_W); otherwise, the algorithm should have selected path P' instead of path P since it has less cost. If they meet at (i_W, j_W), then again, the algorithm should have selected P' instead of P. If they meet at a point (i_s, j_s) before (i_W, j_W), then the cost of P' at (i_s, j_s) should be less than the cost of P up to that point. But then, the cost of P' at (i_W, j_W) should be less than the cost of P which leads to a contradiction. □

4.3 Both shapes are closed

We address this case in [25]. However, that algorithm is not optimal (i.e., may miss the least cost match). In this work, 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 (for each starting point on A), and by taking the least cost match as the cost of matching. Notice that, unlike the algorithm of [25], this algorithm is optimal (i.e., will always find the least cost match) since the algorithm for open and closed shape matching is optimal.

4.4 Complexity

The run-time complexity of our algorithm depends on the time of computing ψ, the cost of matching two sequences of segments. This is the basic operation of the algorithm. From Equation 12, the cost computation at each $cell(i,j)$ takes $\frac{ij}{2}$ time. Therefore, the time complexity for filling a DP table of size $\mathcal{M} \times \mathcal{N}$ is $\mathcal{O}(M^2N^2)$. This is the time complexity of the algorithm when at least one of the shapes is open. If both shapes are closed, the algorithm is repeated M times (i.e., for all starting points of A) so the time complexity of the algorithm becomes $\mathcal{O}(M^3N^2)$. Notice that the complexity of the k-best algorithm of [25] is $\mathcal{O}(kM^3N^3)$ (i.e., the algorithm examines $\frac{MN}{2}$ starting points instead of just M) that is, not only it is not optimal but also, it has higher time complexity.

4.5 Matching examples

Figure 6 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). In each example, one of the two shapes has been shrunk and rotated so as to illustrate the associations between matched parts of the two shapes. Note the parts of the outer (bigger) shapes that do not match parts of the inner shapes and have been left unassociated.

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.

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

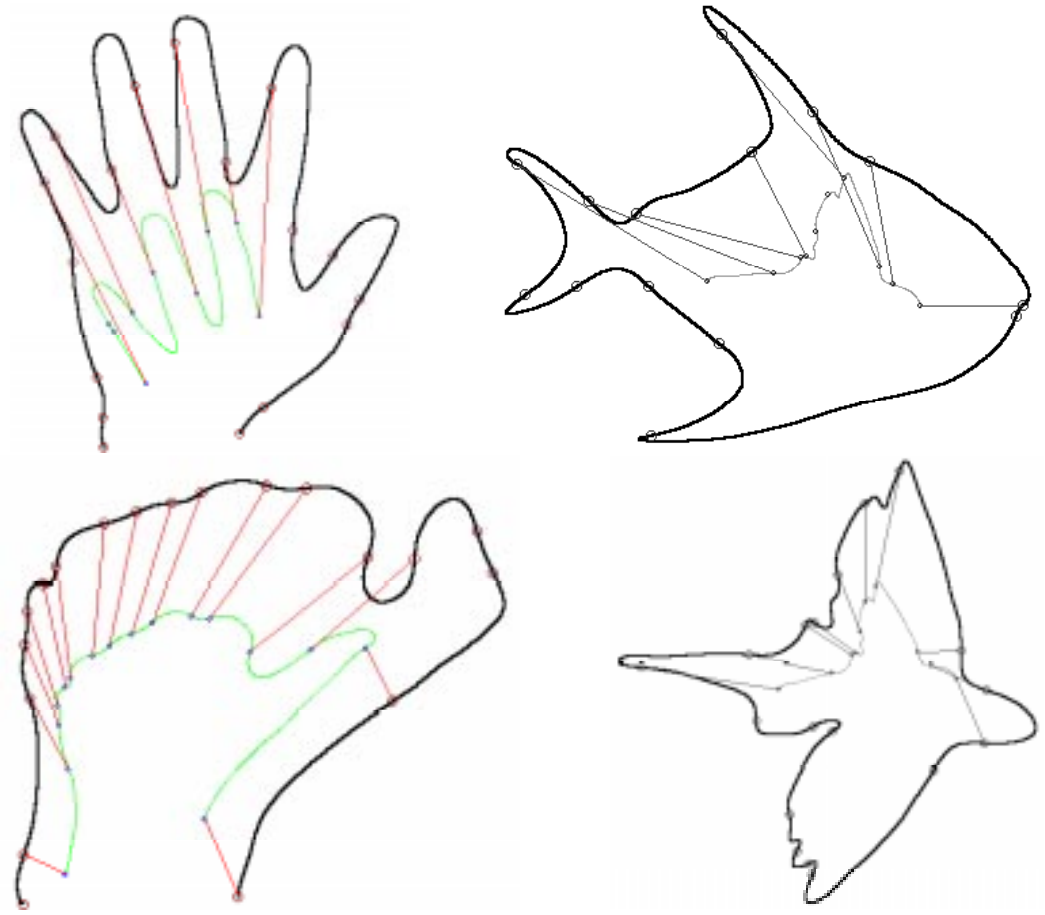

Figure 6: *Segment associations reported by the matching algorithm on representative matches from the gestures and the marine life databases.*

- MARINE[2]: Consists of 1,500 open shapes of marine species which are generated from 1,100 closed shapes[3].

Each shape is represented by its contour. All contours are preprocessed to contain between 80-100 points. The fish silhouettes contain much finer contour detail than the hand silhouettes. To evaluate our algorithm we carefully created 34 queries (17 closed and 17 open shapes) for each dataset. Each query retrieves the best 50 answers.

[2] `http://www.cs.yorku.ca/~eem/marineDB?.`

[3] `http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html.`

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 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.

5.1 Experimental results

Figure 7 illustrates the average values of precision as a function of the size of the answer from 1 to 50. The horizontal axis corresponds to the size of the answer and the vertical axis corresponds to the measured precision. Each dataset is represented by a curve. Each point of each curve is the average over 34 queries. The total number of points in each curve is 50 (i.e., we compute precision for answers containing between 1 and 50 shapes). Therefore, the leftmost point of the diagram corresponds to precision for the best answer (best match), while the rightmost point corresponds to precision for the entire answer set with 50 retrieved shapes.

For small answer sets, both methods achieve precision higher or close to 0.7, that is, almost 70% of the answers are correct. However, precision drops to 0.3 for answers containing 50 shapes. This result demonstrates that our method is best suited for applications where one is interested in retrieving a few best matches. Note 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 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. In particular, retrievals took approximately 10 minutes per query on the GESTURES dataset and approximately 20 minutes per query on the MARINE dataset. Certain optimizations that could speed-up our method are possible, such as the precomputation and storage of the features of the convex and concave segments

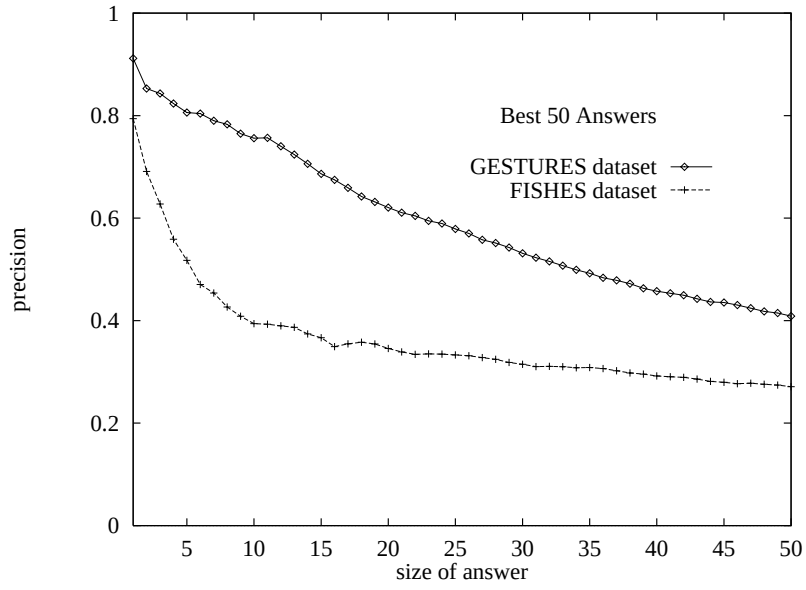

Figure 7: *Average values of precision for the GESTURES and the MARINE datasets as a function of size of the answer set.*

of all shapes in a dataset. In the current implementation these features are computed anew in real-time during each retrieval. In addition, a smoothing on shape contours that reduces the number of inflection points below a small predefined constant (e.g., 20) could speed up time responses significantly.

Certain optimizations that could improve the accuracy of the method are also possible: For example, we can specify different values of the parameter λ of Equation 2 for the two datasets. For example, a value of $\lambda < 1$ favors merging and this could be desirable for shapes with noise and much shape detail such as those of the MARINE dataset.

6 Conclusions

We propose a shape matching algorithm for handling deformed and occluded shape similarity retrievals in image databases. Our algorithm is based on dynamic programming and performs implicitly at multiple scales by allowing segment merges before

association. We demonstrate the effectiveness of our algorithm to shape matching and retrieval using two different datasets with 980 and 1,500 shapes respectively.

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 to Marita Roussou and Panos Economopoulos who implemented parts of the shape matching algorithm in C++. We are also grateful to Prof. Mokhtarian of the Centre for Vision, Speech and Signal Processing laboratory at the University of Surrey, UK, for making available the marine dataset.

References

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

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

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

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

[5] Min-Hong Han and Dongsig Jang. The Use of Maximum Curvature Points for the Recognition of Partially Occluded Objects. *Pattern Recognition*, 23(1/2):21–33, 1990.

[6] Paul G. Gottschalk, Jerry L. Turney, and Trevor N. Mudge. Efficient Recognition of Partially Visible Objects Using a Logarithmic Complexity Matching Technique. *The International Journal of Robotics Research*, 8(6):110–131, December 1989.

[7] Wen-Hsiang Tsai and Shiaw-Shian Yu. Attributed String Matching with Merging for Shape Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):453–462, 1985.

[8] Bir Bhanu and Olivier D. Faugeras. Shape Matching of Two-Dimensional Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):137–156, 1984.

[9] Richard J. Prokop and Anthony P. Reeves. A Survey of Moment-based Techniques for Unoccluded Object Representation and Recognition. *CVGIP: Graphical Models and Image Processing*, 54(5):438–460, 1992.

[10] 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.

[11] J. L. Turney, T. Mudge, and R. Volz. Recognizing Partially Occluded Parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):410–421, 1985.

[12] L. Spirkovska and M. Reid. Robust Position, Scale and Rotation Invariant Object Recognition Using Higher-Order Neural Networks. *Pattern Recognition*, 25(9):975–985, 1992.

[13] G. Papadourakis G. Bebis and S. Orphanoudakis. Curvature Scale Space Driven Object Recognition with an Indexing Scheme based on Artificial Neural Networks. *Pattern Recognition*, 32(7):1175–1201, 1999.

[14] A. Witkin, D. Terzopoulos, and M. Kass. Signal Matching through Scale-Space. In *Proceedings of AAAI-86*, pages 714–719, 1986.

[15] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *Int. J. Computer Vision*, 1(4):321–331, 1988.

[16] B. Vemuri and R. Malladi. Constructing Intrinsic Parameters with Active Models for Invariant Surface Reconstruction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(7), July 1993.

[17] F. Leymarie and M. Levine. Tracking Deformable Objects in the Plane using an Active Contour Model. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(6), 1993.

[18] Alberto Del Bimbo and Pietro Pala. Visual Image Retrieval by Elastic Matching of User Sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, February 1997.

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

[20] E. Saund. Symbolic Construction of a 2D Scale-Space Image. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 817–830, August 1990.

[21] K. Siddiqi and B. Kimia. Parts of Visual Form: Computational Aspects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(3):239–251, 1995.

[22] 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.

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

[24] 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.

[25] Euripides Petrakis and Evangelos Milios. Efficient Retrieval by Shape Content. In *IEEE Multimedia Systems '99*, pages 616–621, Florence, Italy, June 1999. vol. 2.

[26] D. Geiger, A. Gupta, L. Costa, and J. Vlontzos. Dynamic Programming for Detecting, Tracking, and Matching Deformable Contours. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(3) and 18(5):294–302, March and May 1995 and 1996.

[27] L. Floreby. A multiscale algorithm for closed contour matching in image sequence. In *IEEE Int. Conf. on Pattern Recognition*, pages 884–888, 1996.

[28] H. Tagare. Deformable 2D Template Matching using Orthogonal Curves. *IEEE Trans. on Medical Imaging*, 16(1):108–117, February 1997.

[29] L. Shapiro. A Structural Model of Shape. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2, March 1980.

[30] D. Wuescher and K. Boyer. Robust Contour Decomposition using a Constant Curvature Criterion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(1):41–51, 1991.

[31] J. Gorman, R. Mitchell, and F. Kuhl. Partial Shape Recognition using Dynamic Programming. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(2):257–266, March 1988.

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

[33] R. Mehrotra and W. Grosky. Shape Matching Utilizing Indexed Hypotheses Generation and Testing. *IEEE Trans. on Robotics and Automation*, 5(1):70–77, February 1989.

[34] N. Ansari and E. Delp. Partial Shape Recognition: A Landmark-Based Approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(5):470–483, May 1990.

[35] M. Leyton. A Process Grammar for Shape. *Artificial Intelligence*, 34:213–247, 1988.

Contents

1 Introduction **2**

2 Related Work **4**

3 Background **5**

3.1 Shape preprocessing 6

3.2 Basic notations 6

3.3 Matching cases 7

3.4 The Dynamic Programming (DP) table 8

3.5 Cost of matching 10

3.6 Scale factor 12

3.7 Dissimilarity cost 13

3.8 Merging cost 14

4 The algorithm **14**

4.1 Outline 14

4.2 Optimality 16

4.3 Both shapes are closed 17

4.4 Complexity 18

4.5 Matching examples 18

5 Shape retrieval **18**

5.1 Experimental results 20

6 Conclusions **21**