Deformed Shape Matching using Multiscale Dynamic Programming

Jyoti Baid, Evangelos Milios
Department of Computer Science
York University
North York, Ontario, Canada, M3J 1P3
eem@cs.yorku.ca

Abstract

Matching a deformed instance of a shape to a prototype is an open problem. In this paper, we present two matching methods for deformed shape, both based on a multilevel dynamic programming technique proposed by Ueda and Suzuki. The first method is a generalization of this search approach from a best-only to a k-best formulation. We demonstrate that this generalization corrects failures of the best-only formulation of Ueda and Suzuki in several instances. The second method does not constrain search by the scale-space evolution of the shape, but only through the imposition of carefully defined cost functions. We demonstrate that this method performs equally well with the scale-space version, while avoiding the heavy computational cost of computing the scale-space representation. We illustrate the application of the above techniques to real two-dimensional static hand gesture data.

1. Introduction

Natural shapes [Saund90, Leyton88] are rarely rigid or describable by a small set of transformation parameters [Grimson90]. They are typically characterized by smooth boundaries and continuous variations, making it difficult to apply to them feature-based techniques [Stein92]. Similarity of natural shapes is not easy to capture computationally, in spite of the human ease in visually identifying it. The problem of matching deformed shape has recently acquired renewed importance, in the search for more sophisticated ways to interact with computers [Wellner 93, Kuch94].

Several representations have been proposed for describing and matching deformed shapes. Different forms of scale space descriptions have been proposed [Saund90, Siddiqi95, Witkin83, Mokhtarian92]. One class of 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 [Witkin86, Vemuri93, Leymarie93]. Another class of matching methods relies on symbolic entities extracted from the low level shape [Shapiro80, Milios89, Wuescher91, Ueda93, Mokhtarian95]. In [Ueda93] a sophisticated dynamic programming (DP) algorithm was described, which can group segments together in order to come up with appropriate correspondences. For example, if one shape is slightly noisier than the other, it is possible for a single convex segment to be broken up into a sequence of two convex segments with a concave segment between them.

The algorithm in [Ueda93] is capable of combining the three segments in the noisier shape and associating them with the single segment of the less noisy shape. The algorithm is a significant improvement over earlier methods, for example [Milios88], that employ techniques outside the framework of dynamic programming for dealing with this problem. The algorithm of [Ueda93] uses the scale space representation of [Witkin83] to constrain the possible merges, i.e. it accepts merges that are only present at coarser scales of the scale space representation.

In this paper, we have extended the [Ueda93] algorithm in substantial ways.

First, the original algorithm performs a best-only search as it looks for minimum-cost paths in the dynamic programming framework. We have identified instances, in which a best-only search strategy may fail to find a valid match between two shapes, although one exists. To address this deficiency, we have extended the dynamic programming algorithm to perform k-best search, and we have demonstrated experimentally that for a small k, the additional space and time requirements are modest, and the algorithm can solve matching problems where the original one fails.

Second, we have implemented a different set of cost measures from the original algorithm, and we have demonstrated improved performance with them. We present an intuitive justification for the new cost measures, and we show experimental results.

Third, we propose that the scale space restriction be removed from the original algorithm. The scale space computation has two drawbacks. The first drawback is that it tends to diffuse the effects of a feature far away from its location as coarser and coarser scales are considered. As a result, the locality of the features is lost at coarser scales. The second drawback is that it is computationally expensive. We present a formulation of the algorithm that does not use scale space to restrict search for segment merges, and we demonstrate that the

results are comparable to those of the original formulation.

This paper is organized as follows. Section 2 describes the generalized DP algorithm for shape matching. Section 3 describes the shape dissimilarity cost equations. Section 4 presents comparative experimental results.

2. The *k*-option Dynamic Programming

This section describes the multi-segment matching of two shapes where a list of groups of segments of one shape is matched with a list of groups of segments of another shape. We find the least cost matching by using a dynamic programming (DP) technique, where a dynamic programming 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 dissimilarity cost. A complete match is a correspondence between groups of consecutive segments in order, such that no segments are left unassociated. The method involves building various paths in the DP table matching groups of segments of each shape and finally choosing the path that leads to a complete match of both shapes with the minimum cost. We now proceed to describe the method in detail.

Definitions. Assume that the two shapes to be matched are *shapeA* and *shapeB*. A *segment* is a part of a contour between two consecutive inflection points, and it is either convex or concave. Let the number of segments in *shapeA* and *shapeB* at the finest scale be N and M respectively. Elements of *shapeA* and *shapeB* will be indexed by i and j respectively. Inflection points are denoted by p_i and p_j. If a scale space representation is used, we add the superscript h or k to denote the scale for *shapeA* or *shapeB* respectively.

$A^{(h)}$ and $B^{(k)}$ are the convex/concave segment sequences of segments a_i^h and b_j^k at scale h and k of *shapeA* and *shapeB*, of length N^h and M^k respectively. a_i^h is a segment between two consecutive inflection points p_i^h and p_{i+1}^h. $a(i\text{-}n \mid i)$ is the sequence of segments $a_{i-n}, a_{i-n+1},...,a_i$ at the finest scale. Similarly for $b(j\text{-}m \mid j)$.

The DP table is a table the columns of which are indexed by i, and its rows by j. In the DP table a link of cell $(i_W\text{-}1, j_W\text{-}1)$ with (i_W,j_W) denotes the matching of segments $a(i_{W-1}+1|i_W)$ with $b(j_{W-1}+1|j_W)$.

All symbols which involve scale h/k when used without the subscript correspond to the finest scale.

A path in the DP table is a sequence of cells, not necessarily adjacent. It represents the matching of groups of segments of *shapeA* with groups of segments of *shapeB*, in sequential order. A path is a linked sequence of cells $(i_0,j_0),(i_1,j_1),...,(i_w,j_w)$, indicating a partial match. A complete match has $(i_0,j_0)=(iw,jw)$.

The function, $\psi(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w))$ represents the the dissimilarity cost between its two arguments.

The term *options* is used for the number of k best choices stored at each cell of the dynamic programming table.

The data structures used by the method are the DP table, and the G_n and G_m tables.

The Table Definitions. The DP table has $2N$ columns and $2M$ rows, corresponding to segments of *shapeA* and *shapeB* respectively repeated twice (to force the algorithm to consider all possible relative rotations between the two shapes). The cell at the intersection of column i and row j is referred to as *cell(i,j)*. Each cell contains four arrays, *g[.]*, t_1*[.]*, t_2*[.]*, and *index [.]*. Index k_w below varies from 0 to K^2-1 for each of the arrays and refers to the (k_w+1)st path at $cell(i_w,j_w)$. Array $g[k_w]$ contains the total dissimilarity cost. Arrays $t_1[k_w]$ and $t_2[k_w]$ contain the number of unmatched segments in *shapeA* and *shapeB* respectively. Finally, array $index[k_w]=k_{w-1}$ is the back link to the previous cell for that path, and is used to retrace the path back from a given $cell(i_w,j_w)$ and an option k_w.

We use the notation $g[i_w,j_w,k_w]$ to denote the value of g for option k_w of $cell(i_w,j_w)$.

The DP table consists of two distinct areas, the initial value area (the left half of the table) and the calculation area (the right half). In the initial value area all g terms are initialized to zero, t_1 to N and t_1 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. All paths of length greater than zero lie in the calculation area. A path is complete when its corresponding t_1 and t_2 both become zero simultaneously.

G_n and G_m are tables of size $2N \times 2M \times K$ that are used, together with the *index* array above, to reconstruct a path for the k_w option at $cell(i_w,j_w)$. Thus if $G_n(i_w,j_w,k_w)=n_w$ and $G_m(i_w,j_w,k_w)=m_w$, then the previous cell (i_{w-1},j_{w-1}) for option k_w of $cell(i_w,j_w)$ is equal to $(i_w\text{-}2n_w+1, j_w\text{-}2m_w+1)$.

The Algorithm. The main idea in the algorithm is to fill the DP table cells and then search for the optimal complete path. Filling the cells involves mainly the computation of the K^2 elements of cost array *g[.]*, and choosing the k best of them to propagate further. The value at a single element of cost array *g[.]* reflects the cost of the partial match represented by the cell and option associated with the element of *g[.]*. All arithmetic operations on the DP table cell index i and j are modulo N and M respectively. The complexity of the algorithm is due to the mechanisms required to

select the K^2 best extensions of previous partial matches(paths) that lead to the current cell. The following is an outline of the algorithm:

```
for  i_w = N+1,  N+2, ..., 2N
  for j_w = 1, 2, ..., 2M do
    fill cell(i_w,j_w)
    check if a complete path has been found
  end for
end for
select the complete path with the lowest cost
retrace corresponding shape match.
```

The `for` loop for j_w does not run over all the indicated values, as convex to concave matches are not possible.

Filling $cell(i_w,j_w)$ of the DP table

To describe the computation, we need the following auxiliary functions:

-$xmin(list,x)$ extracts the x smallest terms from *list*.

-$listat(i_w,j_w,n_w,m_w)$ is the list of the costs of all options at $cell(i_w\text{-}2n_w\text{-}1,j_w\text{-}2m_w\text{-}1)$, augmented by the dissimilarity cost of extending them, $\psi(a(i_{w\text{-}1}+1|i_w), b(j_{w\text{-}1}+1|j_w))$.

-$fmin(f, i)$ returns the integer pair (n,m) that leads to the i-th smallest result in the evaluation of a given function $f(n, m)$.

The function f we need for filling the k_w option of $cell(i_w,j_w)$ is

$f(n,m) = G+\Psi$, where

$G(n,m) = g(i_w\text{-}2n\text{-}1,j_w\text{-}2m\text{-}1,k_w)$

$\Psi(n,m) = \psi(a(i_w\text{-}2n|i_w), b(j_w\text{-}2m|j_w))$

With the above definitions, we fill the cost array g at cell(i_w,j_w) with the K^2 values computed by the loop:

```
for x from 0 to K-1 collect
    xmin (listat (i_w, j_w, fmin ( f, y)) , K)
```

Function $fmin(f, i)$ searches for n's over the range $[0, \frac{N-1}{2}]$ and for m's over the range $[0, \frac{M-1}{2}]$. Additional constraints on acceptable pairs (n, m) are that either $t_1=t_2=0$ (a complete match has been found), or $t_1>0$, $t_2>0$ (ensuring that there exist unmatched segments in both shapes).

A final constraint is the *curvature scale-space constraint* on (n,m) is that $a(i_w\text{-}2n|i_w)$ and $b(j_w\text{-}2m|j_w)$ must actually merge in the scale space representation at some scale h and k respectively.

From among the K^2 options thus computed, we select the K best for further extension.

Additional bookkeeping work involves updating the back links stored in array index and matrices G_n and G_m for the K best partial paths just computed.

The algorithm used in [Ueda 93] is a special case of the K-option DP algorithm just presented with $K=1$.

Retracing the match corresponding to the optimal path. The match corresponding to the optimal path is retraced by following the back links stored in matrices G_n and G_m and in the *index* arrays of the DP table cells.

Limitations of the Original Method of [Ueda93].

The motivation for introducing the k-option method is that, by the use of 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, and are unusable. Also, the best choice at *cell (i, j)* may lead to an expensive match of the remaining unmatched segments. This point is illustrated in the following.

Example. Figure 1 shows the bottom half of the DP table for matching *shapeA* and *shapeB* with 8 features each. Cells in the table are marked as follows: *(5,1)* as B, *(7,1)* as C, *(10,4)* as E, *(11,7)* as A and *(13,9)* as D. At each cell the number in bold represents the cost of the path up to that cell using 1 option (solid line) and the other number the cost along the alternate path (K=2, dotted line).

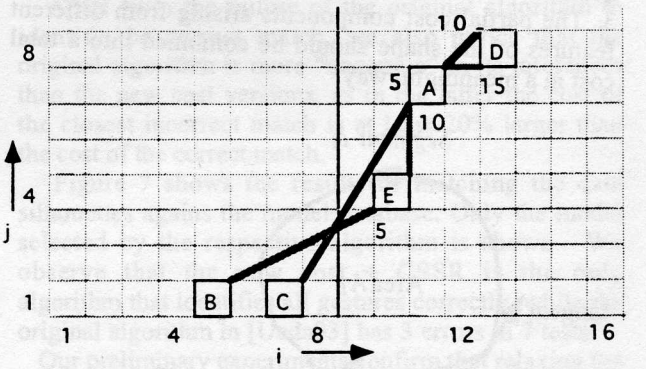


FIGURE 1. Example illustrating failure of the matching using the 1-option DP method.

Alternate path BEAD is ignored by the 1 option algorithm because of a least cost choice made at A, while the selected path CAD is incomplete.

This problem is solved 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.
Examples of this kind of failure of the 1-option algorithm in a real situation of matching the silhouettes of two hand gestures is shown in [TR95]. Two additional optimizations of the 1-option algorithm as presented in [Ueda 93] are also described in [TR95].

3. Determination of Cost Components

In the description of the algorithm we mentioned the cost term $\psi(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w))$, which represents the cost of associating segments $a(i_{w-1}+1|i_w)$ of *shapeA* with segments $b(j_{w-1}+1|j_w)$ of *shapeB*. Computation of the cost term must rely on geometric properties of the segments themselves. The cost term consists of three additive components [Ueda93]:
- the cost of merging segments $a(i_{w-1}+1|i_w)$
- the cost of merging segments $b(j_{w-1}+1|j_w)$
- the cost of associating the merged $a(i_{w-1}+1|i_w)$ with the merged $b(j_{w-1}+1|j_w)$ (dissimilarity cost).

Requirements for the cost computation are the following:
1. Merging should follow the process grammar rules [Leyton88, Milios89], i.e. each allowable merging should be a recursive application of the grammar rules CVC -> C and VCV-> V. This is enforced by the DP algorithm.
2. Merging a visually prominent segment (large size, high curvature) into a merged segment of the opposite type should incur a high cost. To specify this requirement, we need to define visual prominence in geometric terms.
3. The partial cost components arising from different features of the shape should be combined into a total cost in a meaningful way.

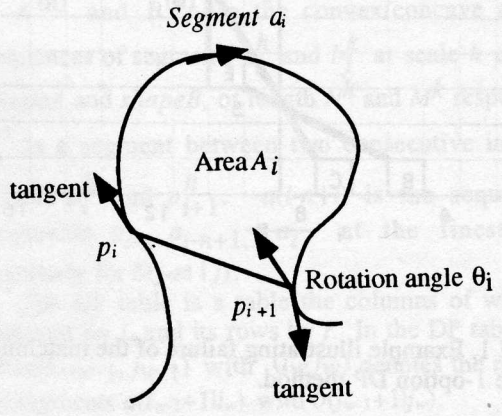


FIGURE 2. Geometric quantities for defining the prominence of a segment.

We now define geometric quantities (features) needed in the specification of visual prominence of a segment a_i (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}. *Length* is the length of segment a_i . *Area* A_i is the area enclosed between the line segment (chord) and the arc between the inflection points, p_i and p_{i+1}.

Dissimilarity cost computation. This cost computation should assign a higher cost to segments (or groups of segments) with large differences in more than one feature. The dissimilarity cost of associating segments $a(i_{w-1}+1|i_w)$ of *shapeA* with segments $b(j_{w-1}+1|j_w)$ of *shapeB* is computed using the following equation:

$$DissimilarityCost = cost \times times$$

The cost is again computed with a max operation over the cost terms corresponding to each feature, for the same reason as described previously.

$$cost = \max_{\text{all features } f} \{ d_f \}$$

We choose the max operation instead of product (as in [Ueda93]) because in the product a small cost in terms of one feature can cancel the effect of a high cost in terms of another, something that may be counter to visual intuition. The max operation addresses this problem.

The term d_f is defined as follows for f being rotation angle.

$$d_\theta = \left| \frac{\Theta_a - \Theta_b}{\Theta_a + \Theta_b} \right|$$

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 *shapeA* and *shapeB* respectively.
For f being length or area,

$$d_f = \left| \frac{f_a}{F_a} - \frac{f_b}{F_b} \right|$$

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*.

Factor *times* is equal to the number of features f for which d_f is greater than $0.75 \times cost$. Thus if all three features have uniformly large d_f, then the dissimilarity cost is multiplied by 3.

Merging cost computation. Let the types of the segments being merged be CVC...VC, where C is convex and V is concave (the opposite case is obtained by switching C and V). The merging cost is defined as follows:

$$MergingCost = \max_{\text{all features } f} \{ c_f w_f \}$$

where subscript f refers to a feature (length, area or rotation angle). We choose the above max operation instead of sum of products of terms comparing consecutive segments (as in [Ueda93]) for two reasons.

The reason for using max instead of product is the same as above. 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 [Ueda93] 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:

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

For f being rotation angle:

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

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]*.

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

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

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

Total cost. This can now be expressed as the following sum:

$$\psi(a(i_{w-1}+1 | i_w), b(j_{w-1}+1 | j_w)) =$$
$$MergingCost\ (\ a(i_{w-1}+1 | i_w)\ ) +$$
$$MergingCost\ (\ b(j_{w-1}+1 | j_w)\ ) +$$

where λ is a constant that represents the relative importance of the merging and dissimilarity costs. In the experiments λ was set to 1.

4. Experimental Analysis and Discussion

In the generalized algorithm presented above, there are several possible choices:

1. whether to use the curvature scale space (CSS) constraint.
2. whether to use the original [Ueda93] or the new [Section 3] cost computations.
3. whether to use one option ($K=1$) or more than one option (k>1).

The algorithm in [Ueda93] corresponds to the use of CSS, the original cost computations, and $K=1$. In the experiments described below, we used $K=5$ to avoid the situations in which the $K=1$ algorithm fails to find a solution, where one exists.

Figure 3 shows a model database of two-dimensional silhouettes of hand gestures, extracted from real video images. The large circles indicate inflection points that are transitions from convex to concave, and the small circles from concave to convex, when the shapes are traversed in a clockwise fashion. Figure 4 shows a set of data silhouettes (belonging to a different person) that are to be matched against the model database. Figures 5 and 6 show the result of matching shape *test_3.s* and *test_8.P.2* respectively against the model database. The models of the database have been sorted by increasing cost, hence the top one is the most similar according to the respective algorithm. The algorithms are the following:

-*[Ueda93]*: the original algorithm with K=5 and with the CSS constraint.

-*[Ueda93]-CSSR:* the original algorithm with K=5 and without the CSS constraint

-*new cost*: the cost terms described in this paper, with K=5 and without the CSS constraint

-*new cost + CSSR:* the cost terms described in this paper, with K=5 and with the CSS constraint.

The cost values shown have been normalized so that the minimum cost for each algorithm is equal to 1.

Apart from the failure of the original algorithm to identify the correct match, we also notice that the original algorithm is more "uncertain" in its inferences than the new cost versions, as in the latter the cost of the closest incorrect match is at least 20% larger than the cost of the correct match.

Figure 7 shows the results of matching the data silhouettes agains the model database. Only the model selected by the respective algorithm is shown. We observe that the new cost + CSSR is the only algorithm that identifies all gestures correctly, while the original algorithm in [Ueda93] has 3 errors in 7 tests.

Our preliminary experiments confirm that relaxing the CSS constraint, while using the new cost measures, improves performance compared with the original cost computations with the CSS constraint. Further investigation of the effectiveness of the proposed algorithm in shape recognition is underway.

References

[TR95] J. Baid, E. Milios. *Deformed Shape Recognition using Dynamic Programming*, Technical

Report, Department of Computer Science, York University, 1995.
[Grimson90] Grimson, W. E. L., *Object Recognition by Computer: The role of Geometric Constraints*, MIT Press, Cambridge, Massachusetts, 1990.
[Kuch94] J. Kuch, T. Huang: "Human-Computer Interaction via the human hand: a hand model", 28th IEEE Asilomar Conf. on Signals, Systems and Computers, 1994, pp. 1252-1256.
[Leymarie93] F. Leymarie, M. Levine: "Tracking deformable objects in the plane using an active contour model", IEEE Transactions on PAMI, 15(6), June 1993.
[Leyton88] M. Leyton: "A process grammar for shape", Artificial Intelligence, 34, 1988, pp. 213-247.
[Milios89] Milios, E. E., "Shape Matching using curvature processes", *Computer Vision, Graphics and Image Processing*, vol. 47, pp. 203-226, 1989.
[Mokhtarian95] Mokhtarian, F. "Silhouette-Based Object Recognition through Curvature Scale Space", IEEE Transactions on PAMI, 17(5), May 1995.
[Saund90] E. Saund: "Symbolic construction of a 2D scale-space image", IEEE Trans. on PAMI, August 1990, pp. 817-830.
[Sclaroff94] S. Sclaroff, A. Pentland: "Search by shape examples: modelling nonrigid deformations", 28th IEEE Asilomar Conf. on Signals, Systems and Computers, 1994, pp. 1341-1345.
[Shapiro80] L. Shapiro: "A structural model of shape", IEEE Transactions on PAMI, Vol. 2, March 1980.
[Siddiqi95] Siddiqi, K. and Kimia, B.B., "Parts of Visual Form: Computational Aspects", *IEEE Trans. on PAMI,* vol. 17, no. 3, pp. 239-251, 1995.
[Stein92] Stein, F. and Medioni, G., "Structured Indexing: Efficient 3-D object Recognition", *IEEE Trans. on PAMI*, vol. 14, pp. 125-145, 1992.
[Ueda93] Ueda, N. and Suzuki, S., "Learning Visual Models from Shape Contours Using Multiscale Convex/Concave Structure Matching", I*EEE Trans. on PAMI,* vol. 15, No. 4, pp. 337-352, April 1993.
[Vemuri93] Vemuri, B.C. and Malladi, R., "Constructing Intrinsic Parameters with Active Models for Invariant Surface Reconstruction", *IEEE Trans. on PAMI*, vol. 15, no. 7, July 1993.
[Wellner93] Wellner, P., "Digital Desk", *Communications of the ACM*, vol. 36, No. 7, 1993.
[Witkin83] Witkin, A. P., "Scale space filtering", *Proceedings 8th IJCAI* (Karlsruhe, West Germany), pp. 1019-1022, 1983.
[Witkin86] Witkin, A., D. Terzopoulos, M. Kass, "Signal Matching through Scale-Space", Proceedings of AAAI-86, pp. 714-719.
[Wuescher91] Wuescher, D., K. Boyer: "Robust contour decomposition using a constant curvature criterion", *IEEE Transactions on PAMI,* 13(1), pp. 41-51, 1991.

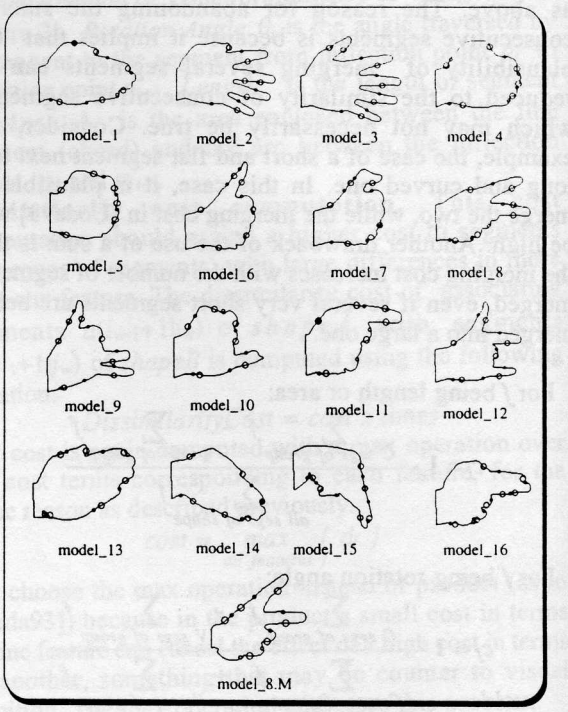


FIGURE 3. Model shapes.

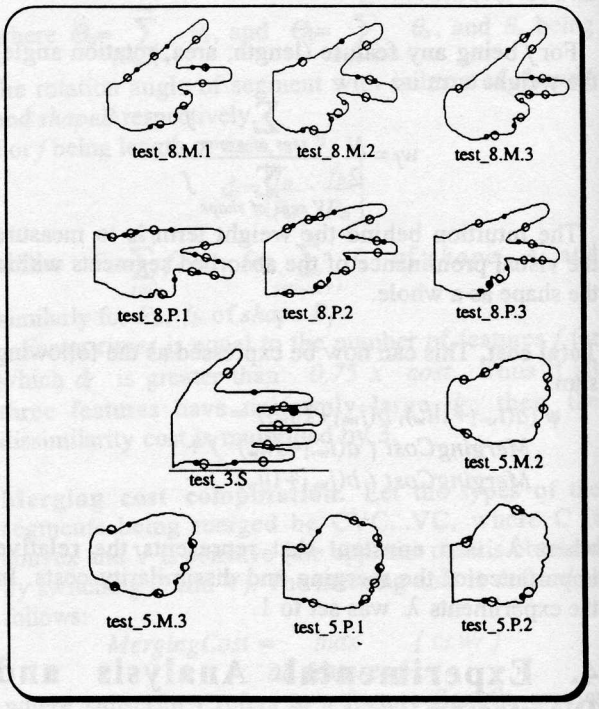


FIGURE 4. Test shapes

test shape to be matched against all shapes in the gesture database, using all cost equation

test_8.P.2

Results Table

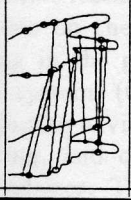
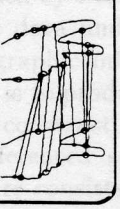
	[Ueda 93]	[Ueda 93] -CSSR	new cost	new cost -CSSR
Results of matching with the database. Sorted on cost, therefore, first model is most similar. Normalized to range [1..100]	model 6 1 model 9 9 model 8.M 30 model 12 48 model 3 55 model 3 64 model 2 66 model 8 69 model 15 80 model 7 81 model 14 86 model 10 92 model 11 94 model 1 95 model 6 95 model 5 99 model 4 100	model 8 1 model 8.M 9 model 11 18 model 9 20 model 6 24 model 2 25 model 2 34 model 13 35 model 4 38 model 5 43 model 16 45 model 10 49 model 15 53 model 7 57 model 3 73 model 1 94 model 14 100	model 8 1 model 8.M 11 model 3 47 model 6 47 model 7 47 model 11 48 model 9 61 model 13 61 model 16 64 model 4 66 model 12 68 model 15 75 model 1 75 model 10 78 model 5 83 model 14 84 model 2 100	model 8 1 model 8.M 21 model 9 40 model 2 42 model 13 45 model 3 49 model 7 51 model 11 53 model 6 56 model 12 56 model 15 59 model 16 67 model 4 78 model 14 82 model 1 86 model 10 95 model 5 100
Matching of *test8.P.1* with *model_8.M*				

FIGURE 5. Matching a test shape to the model database.

test shape to be matched against all shapes in the gesture database, using all cost equations

test_3.s

Results Table

Cost Equation	[Ueda 93]	[Ueda 93] -CSSR	new cost	new cost+CSSR
Results of matching with the database. Sorted on cost, therefore, first model is most similar. Normalized to range [1..100]	model 4 1 model 8.M 2 model 13 2 model 9 2 model 3 6 model 12 8 model 5 22 model 16 25 model 1 46 model 8 49 model 7 54 model 10 61 model 2 64 model 14 77 model 15 82 model 11 100 model 6 100	model 12 1 model 3 3 model 9 14 model 8.M 17 model 8 17 model 13 21 model 10 22 model 11 25 model 2 29 model 7 33 model 15 36 model 5 41 model 5 50 model 4 59 model 14 69 model 16 78 model 1 100	model 3 1 model 12 18 model 8.M 32 model 2 33 model 8 33 model 10 42 model 9 49 model 15 52 model 11 55 model 13 60 model 5 63 model 16 66 model 14 67 model 7 68 model 6 75 model 1 84 model 4 100	model 3 1 model 12 18 model 8.M 28 model 2 51 model 13 61 model 15 64 model 4 72 model 8 75 model 9 75 model 16 76 model 11 83 model 7 84 model 10 90 model 6 90 model 1 95 model 5 98 model 14 100
Matching of *test_3.s* with *model_3*				

FIGURE 6. Matching another test shape to the model database.

	Model selected by algorithm			
test shape (correct model)	Ueda 93	Ueda 93 - CSS	New cost	New cost + CSS
3.S (3)	4	12	3	3
8.M.1 (8, 8.M)	8.M	8.M	8.M	8.M
8.M.2 (8, 8M)	8	8.M	8.M	8
8.M.3 (8, 8M)	8	8.M	7	8.M
8.P.1 (8, 8M)	8M	8M	8M	8M
8.P.2 (8, 8M)	16	8	8M	8
8.P.3 (8, 8M)	4	8M	8	8

FIGURE 7. Results of matching the test shapes against the model database.