In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), New York, June 17-22 (2006).

Hierarchical Procrustes Matching for Shape Retrieval

Graham McNeill, Sethu Vijayakumar
Institute of Perception, Action and Behavior
School of Informatics, University of Edinburgh, Edinburgh, UK. EH9 3JZ
G.J.McNeill-2@sms.ed.ac.uk, sethu.vijayakumar@ed.ac.uk

Abstract

We introduce Hierarchical Procrustes Matching (HPM), a segment-based shape matching algorithm which avoids problems associated with purely global or local methods and performs well on benchmark shape retrieval tests. The simplicity of the shape representation leads to a powerful matching algorithm which incorporates intuitive ideas about the perceptual nature of shape while being computationally efficient. This includes the ability to match similar parts even when they occur at different scales or positions. While comparison of multiscale shape representations is typically based on specific features, HPM avoids the need to extract such features. The hierarchical structure of the algorithm captures the appealing notion that matching should proceed in a global to local direction.

1. Introduction

In many object recognition problems, the classes are most easily disambiguated on the basis of *shape* - as opposed to properties such as color or texture. However, defining a shape descriptor and associated matching algorithm which is both *generic* and *discriminative* is a difficult challenge. Hierarchical Procrustes Matching (HPM) approximates the perceptual similarity between two shapes by matching progressively smaller boundary segments. Longer segments that have already been matched provide initial matches for the shorter segments, which can then slide and stretch/contract in order to find the best matches at this smaller scale. The individual segment matches reflect the similarity of features that are relevant at the current length. Rather than working with segments of a particular type (*e.g.* convex segments), HPM essentially approximates a mapping which takes any segment of one shape to its counterpart on the other shape. In the next section we briefly review the relevant literature and explain the motivation behind HPM. The algorithm is described in detail in Sec. 3 and evaluated on a range of benchmark data sets in Sec. 4.

2. Motivation and Related Work

Simple, global point matching methods [20, 14] work surprisingly well in many shape retrieval and classification tasks. These find a single linear transformation which best aligns one point set with the other. This approach fails when more complex transformations are present such as independent movement of parts or smooth deformations (Fig. 1). To deal with such cases, a more local or nonlinear approach is needed (*e.g.* [5, 17]). However, techniques which focus on local shape differences suffer from other problems. For example, the curve alignment algorithm of Sebastian *et al.* [17] remains effective in the presence of smooth deformations and, to some extent, part articulation. However, the authors note that their algorithm regards a handwritten "6" and "U" as similar, since the curvature of the two shapes only differs at a few points.

The above discussion indicates that even in apparently simple cases, there can be a large discrepancy between a perceived difference in shape and the approximation of this difference from a global or local perspective. It therefore seems natural to investigate whether a generic shape matching algorithm should lie somewhere between these two extremes – a neighborhood-based approach. More generally, one can imagine an algorithm that is not restricted to comparisons based on a single neighborhood size. We realize this idea by considering segments of different lengths. HPM is a multiscale approach that investigates shape matching at a variety of different positions (segment start points) and scales (segment lengths). In shape matching, 'scale' can relate to a number of related ideas: frequency (*e.g.* wavelets [6]), the degree of filtering that the shape has undergone [7, 16], or the resolution at which the shape is sampled [2]. Here, the notion of scale arises through the matching process (rather than just the shape representation itself) since the least squares criteria used to assess segment similarity is dominated by the 'global' shape of the segment.

Many techniques match shapes based on specific features, such as points of zero curvature [16], points of minimum curvature [7], and convex/concave segments [15, 11].

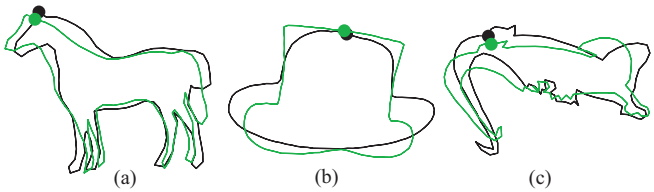


Figure 1. Examples of shapes matched using GPM (linear interpolation between points is used for clarity). The filled circles show corresponding points. In all cases, the correct correspondence is found, but a good alignment is not possible.

In contrast, HPM compares segments explicitly – rather than comparing very specific features, and does not restrict itself to working with segments of a particular type.

Corresponding regions of shapes often appear at slightly different scales and positions. For example, the dorsal fin of a fish may be both larger and further forward than its counterpart on a second fish. Matching algorithms must be flexible enough to allow for this phenomenon. HPM achieves this by matching in a global to local direction. Longer segments that have already been matched provide initial matches for the shorter segments, which can then slide and stretch/contract in order to find the best matches at this smaller scale. In the context of our example, if we have already aligned the backs of the two fish, it means that we have an approximate alignment for the fins. We then search in the neighborhood of this estimate (i.e. over segments with start points and lengths close to this estimate) to find the best match. To prevent the algorithm matching fins that are too far apart (the more candidate segments that are considered, the greater the chances of finding a spurious match), we can penalize deviations from the initial prediction (Sec. 3.4.2), or simply use the neighborhood size (Sec. 3.4.1). We now describe the HPM algorithm in detail.

3. Hierarchical Procrustes Matching (HPM)

The first step is to specify how two shapes are compared when the correct correspondence is known. In this paper we use Procrustes matching, the main concepts of which are summarized below.

3.1. Procrustes Matching

Assume that we have two objects, each represented by n 2D points: $\mathbf{U} = (\mathbf{U}_1, \ldots \mathbf{U}_N)^T$, $\mathbf{V} = (\mathbf{V}_1, \ldots \mathbf{V}_N)^T \in \mathbb{R}^{N\times 2}$, and that the point-to-point correspondence is known to be $\mathbf{U}_1 \leftrightarrow \mathbf{V}_1, \mathbf{U}_2 \leftrightarrow \mathbf{V}_2, \ldots$ Intuitively, the shape of an object should not change under translation, rotation or scaling, so a reasonable measure of similarity between $\mathbf{U}$ and $\mathbf{V}$ is the minimum sum of squared distances over corresponding point pairs:

$$d_P(\mathbf{U}, \mathbf{V}) \equiv \min_{a,\mathbf{t},\Gamma} \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{V}_n - (a\Gamma\mathbf{U}_n + \mathbf{t})\|^2, \quad (1)$$

where a is a scale parameter, $\mathbf{t} \in \mathbb{R}^2$ is a translation vector and Γ is a 2D rotation matrix. We refer to the value $d_P(\mathbf{U}, \mathbf{V})$ as the *Procrustes distance* (PD). It is common to use a normalized version of the PD where points sets are first scaled to have unit size [8]. In this paper, we will normalize the *boundary length* of the polygon associated with $\mathbf{V}$ and then transform $\mathbf{U}$ to match $\mathbf{V}$, so the PD used here is not symmetric in its arguments. For 2D shapes, $d_P(\mathbf{U}, \mathbf{V})$ can be calculated using a simple closed form expression as follows [8]. Represent each 2D point as a complex number, $\mathbf{V}_n = (x_n, y_n) \rightarrow x_n + iy_n = w_n \in \mathbb{C}$. Then, $\mathbf{V} \rightarrow \mathbf{w}$ and $\mathbf{U} \rightarrow \mathbf{z} \in \mathbb{C}^N$. After centering $\mathbf{w}$ and $\mathbf{z}$ so that $\sum_{n=1}^{N} w_n = \sum_{n=1}^{N} z_n = 0$, the PD between $\mathbf{U}$ and $\mathbf{V}$ is given by

$$d_P(\mathbf{U}, \mathbf{V}) = (\mathbf{w}^*\mathbf{w} - (\mathbf{w}^*\mathbf{z}\mathbf{z}^*\mathbf{w})/\mathbf{z}^*\mathbf{z})/N. \quad (2)$$

The PD is used to measure the similarity of segments (or entire shapes) throughout this paper.

3.2. Global Matching

Given two shapes $\mathbf{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_N)^T$ and $\mathbf{V} = (\mathbf{V}_1, \ldots \mathbf{V}_N)^T$, we find a global correspondence between them using the technique described in [14]. This finds a 1-1 correspondence between the two point sets, under the assumption that the cyclic ordering of the indices is respected by the correspondence. For example, $\mathbf{U}_{12} \leftrightarrow \mathbf{V}_{78}, \mathbf{U}_{13} \leftrightarrow \mathbf{V}_{79}, \ldots$ is a valid correspondence. There are N valid correspondences. We compute the N PDs associated with these and the N PDs associated with the reflection of $\mathbf{U}$, and then select the correspondence with the smallest PD. This technique is referred to as Global Procrustes Matching (GPM) throughout. The global alignments shown in Fig. 1 were found using this approach. Note that GPM is very fast since the number of correspondences to check is linear in N, and PDs can be computed efficiently (eq.(2)). The PD associated with GPM can be used as similarity score between shapes and has been shown to be moderately effective in shape retrieval tests [14].

HPM requires a roughly correct global alignment, and it may seem somewhat risky to use a technique as simple as GPM to find this alignment. However, the high performance of HPM on retrieval tasks (Sec. 4) (and in particular, the increase in performance over GPM) suggests that GPM generally *does* find the correct correspondence – even with difficult examples (Fig. 1). The problem with GPM is that the PD does not approximate perceptual similarity accurately enough to retrieve the correct shapes.

Algorithm 1 Pseudocode for HPM.

1: Initialize: match $\mathbf{U}$ and $\mathbf{V}$ using GPM
2: **for** segment lengths: $l = 50, 25, 12.5$ **do**
3: **for** segments of $\mathbf{U}$: $m = 1, \ldots,$ # segs. at len. l **do**
4: compute the *predicted match* of seg. m based on the matches at length l-1
5: define neighborhood around the *predicted match*
6: compute PD between seg. m and each neighborhood seg. of $\mathbf{V}$
7: **end for**
8: Select matches and get total score for len. l based on:
i. PD between segs. and candidate matches
ii. deviation from match predicted by l-1
iii. deviation from match predicted by neighbors
9: **end for**
10: shape similarity = wtd. sum of scores at each length

3.3. Segment Representation

We assume that $\mathbf{U}$ and $\mathbf{V}$ are 2D shapes, each represented by N ordered points taken at equal intervals along their respective closed boundaries. The shapes are normalized so that the polygons formed by their points have boundary length equal to 100. A global correspondence between $\mathbf{U}$ and $\mathbf{V}$ is found using GPM (Sec. 3.2) and points are relabeled such that $\mathbf{V}_1$ corresponds to $\mathbf{U}_1$. A segment of the $\mathbf{U}$ polygon is denoted by $\mathbf{U}(s_U, l_U)$, where s_U is the segment's start point and l_U its length. A segment is represented by k equally spaced points taken from the relevant segment of the complete polygon. Note that a segment of any length and start point (*i.e*. s_U and l_U need not be natural numbers) can be represented by any number of points, k, by re-sampling from the polygon in this way. A segment of length l is represented by $k \simeq \frac{l}{100}N$ points – *i.e*. the original sampling frequency of the boundary is approximately maintained.

To match $\mathbf{U}$ to $\mathbf{V}$, we consider the full shape $\mathbf{U}$, four segments of length 50, eight of length 25 and sixteen of length 12.5. At each length, neighboring segments overlap by one half. Each of these segments will, at some stage, be compared to multiple segments of $\mathbf{V}$ which have different start points and lengths. If the segment of $\mathbf{U}$ has k points, the segments on $\mathbf{Y}$ it is compared to will also be represented by k points, enabling the PD between segments to be computed – the points are indexed in order of their position on the segment so the correspondence is known.

3.4. Hierarchical Matching

Having described how the similarity of any pair of k-point shapes is computed (Sec. 3.1), how the global match is found (Sec. 3.2), and how segments are represented (Sec. 3.3), we are now in a position to describe the HPM algorithm (summarized in Algorithm 1). Two variants of HPM are considered. The first prevents invalid segment matches by limiting the permitted change in start point and length. The second uses a softer approach which requires dynamic programming.

3.4.1 Fast Method

Consider the task of matching the segment of $\mathbf{U}$ of length 50 and start point 0, $\mathbf{U}(0, 50)$. The global correspondence 'predicts' that $\mathbf{U}(0, 50)$ should be matched to $\mathbf{V}(0, 50)$. As discussed in Sec. 2, the segment $\mathbf{U}(0, 50)$ should be allowed to stretch/contract and slide along the boundary of $\mathbf{V}$, but only to a limited extent. Formally, $\mathbf{U}(0, 50)$ must be matched to a segment $\mathbf{V}(0 \pm \delta_s, 50 \pm \delta_l)$, where the maximum values of δ_s and δ_l determine the permitted slide and stretch/contraction, and the sums in the arguments are taken modulo 100. In practice, we define a 7×7 grid centered at $(0, 50)$ and compute the PD between $\mathbf{U}(0, 50)$ and the segments of $\mathbf{V}$ associated with the grid points (Alg. 1, steps 4-6). Segment $\mathbf{U}(0, 50)$ is then matched with whichever segment of $\mathbf{V}$ is closest in terms of the PD (Alg. 1, step 8 – only 8(i) is used here; 8(i)-(iii) are used for the extended method described in Sec. 3.4.2). The same approach is used to match the other segments of length 50. The matched segments of length 50 provide initial estimates for matching the segments of length 25 and so on. The maximum permitted slide and stretch decreases with segment size. In our experiments, the segments of length 50 were allowed to stretch or contract by a maximum of 3% (with increments of 1% being evaluated) and to shift along the boundary by $\pm1.5\%$. For segments of length 25, these values were halved, and so on.

The similarity of $\mathbf{U}$ and $\mathbf{V}$ is a weighted sum of the PDs over all the matched segments. The PD is normalized for the number of points (eq.(2)), so the PDs corresponding to different length segments are of a similar magnitude (recall that the segments of length 50 have $\sim N/2$ points whereas the segments of length 25 have $\sim N/4$ points). However, there is less variation between shorter segments (*c.f*. Fig. 2) so the PDs are generally somewhat smaller. We ensure that matches at every length make a roughly equal contribution to the final similarity score by giving a higher weight to PDs from shorter segments. Let S_l denote the sum of PDs over the matched segments at a fixed length l. The *asymmetric similarity* of $\mathbf{U}$ and $\mathbf{V}$ is given by

$$d_F(\mathbf{U}, \mathbf{V}) \equiv w_{100} d_P(\mathbf{U}, \mathbf{V}) + \sum_{l=50,25,12.5} w_l S_l \qquad (3)$$

where the w_l are constant weights. The weights used for the evaluations in Sec. 4 were determined experimentally, and the same weights were used for all data sets. Since $d_F(\mathbf{U}, \mathbf{V}) \neq d_F(\mathbf{V}, \mathbf{U})$, we take the (symmetric) *similarity*

of $\mathbf{U}$ and $\mathbf{V}$ to be

$$D_F(\mathbf{U},\mathbf{V}) \equiv d_F(\mathbf{U},\mathbf{V}) + d_F(\mathbf{V},\mathbf{U}). \quad (4)$$

Note that $d_F(\mathbf{U},\mathbf{V})$ (eq.(3)) depends on the position of s_U=0 on the polygon $\mathbf{U}$. We have not yet investigated whether there is any significant variation in performance associated with this choice.

3.4.2 Dynamic Programming Approach

This section describes a more sophisticated technique for selecting the segment-segment matches. Readers looking for a brief introduction to HPM can skip straight to the evaluations in Sec. 4 and return to this section later if they wish. In Sec. 3.4.1, the valid segment mappings were determined by a hard limit on the amount that a segment of $\mathbf{U}$ could slide and stretch away from its predicted match on $\mathbf{V}$. Here, we use a softer approach, whereby the amount of slide and stretch is penalized. Consider a segment $\mathbf{U}(s_U, l_U)$, and assume that its *predicted* match on $\mathbf{V}$ (given the already matched longer segments) is $\mathbf{V}(s_V^p, l_V^p)$. If the *selected* match is $\mathbf{V}(s_V, l_V)$, then we can penalize the deviation from the predicted match using the value $|(s_V, l_V) - (s_V^p, l_V^p)|$. Rather than selecting a segment $\mathbf{V}(s_V, l_V)$ purely on the basis of PD, we now select it using

$$\mathbf{V}(s_V, l_V) = \arg \min_{\mathbf{V}(s_V, l_V)} d'_P(\mathbf{U}(s_U, l_U), \mathbf{V}(s_V, l_V)) \quad (5)$$

where

$$\begin{aligned} d'_P(\mathbf{U}(s_U, l_U), \mathbf{V}(s_V, l_V)) \equiv \; & d_P(\mathbf{U}(s_U, l_U), \mathbf{V}(s_V, l_V)) \\ + \; & \lambda_{l_U} |(s_V, l_V) - (s_V^p, l_V^p)|. \end{aligned} \quad (6)$$

The parameter λ_{l_U} specifies the importance of staying close to the predictions at length l_U. The λ_l's should be chosen so that the penalty terms have a similar impact at each level of matching. The values used for the evaluations in Sec. 4 were determined experimentally, and the same values were used for all data sets.

We may also want to encourage consistency between the matches of neighboring segments of the same length – bearing in mind that these overlap by one half. For example, if the segment $\mathbf{U}(0, 50)$ is slid far to the left of its predicted match on $\mathbf{V}$, it seems inappropriate that the neighboring segment $\mathbf{U}(25, 50)$ should slide to the right. Just as the longer segments predict a match for shorter segments, we think of a segment $\mathbf{U}(s_U, l_U)$ predicting matches for its neighbors on either side. Specifically, if $\mathbf{U}(s_U, l)$ is matched to a segment $\mathbf{V}(s_V, l_V)$, it predicts that its neighbors of the same length, $\mathbf{U}(s_U \pm l/2, l)$, are matched to $\mathbf{V}(s_V \pm l_V/2, l_V)$.

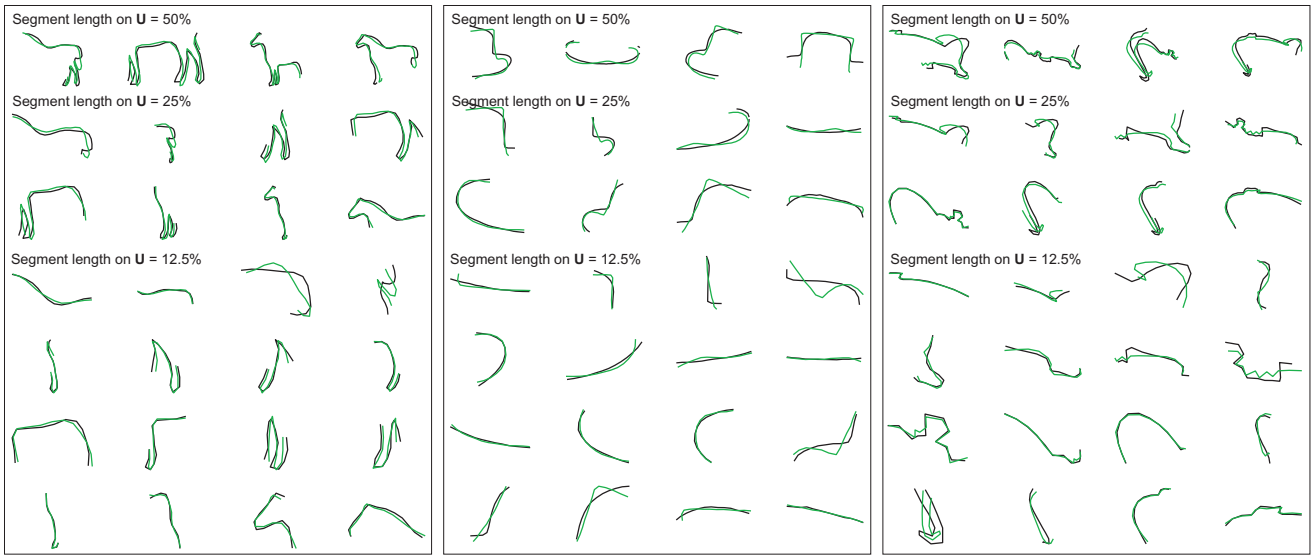
Given a segment $\mathbf{U}(s_U, l_U)$, the segment $\mathbf{V}(s_V, l_V)$ that it is matched to will now depend on the PD between the segments, the penalty for choosing $\mathbf{V}(s_V, l_V)$ given the prediction from the already matched longer segments, and an additional penalty associated with the neighbors of $\mathbf{U}(s_U, l_U)$ of the same length. Using this new penalty is problematic since the neighbors of $\mathbf{U}(s_U, l_U)$ have *not* already been matched. Unlike the way that a global match provides predictions for the segments of length 50 and so on, there is no natural starting place to begin matching when considering segments of the same length – we must effectively choose all the matches simultaneously. If the M segments of $\mathbf{U}$ of a fixed length l (*e.g.* there are M=4 segments of length 50) are denoted by $\mathbf{U}(s_{U_1}, l), \ldots, \mathbf{U}(s_{U_M}, l)$, then the segments of $\mathbf{V}$ that they are matched to: $\mathbf{V}(s_{V_1}, l_{V_1}), \ldots, \mathbf{V}(s_{V_M}, l_{V_M})$, are chosen so as to minimize

$$\begin{aligned} \mathcal{J} \equiv & \sum_{m=1}^{M} \{ d'_P(\mathbf{U}(s_{U_m}, l), \mathbf{V}(s_{V_m}, l_{V_m})) \\ + & \frac{\lambda_l}{2} |(s_{V_m}, l_{V_m}) - ((s_{V_{m-1}} + l_{V_{m-1}}/2), l_{V_{m-1}})| \\ + & \frac{\lambda_l}{2} |(s_{V_m}, l_{V_m}) - ((s_{V_{m+1}} - l_{V_{m+1}}/2), l_{V_{m+1}})| \}, \end{aligned}$$

where d'_P and λ_l are from eq.(6), and the sums m+1 and m-1 are taken modulo M. As in Sec. 3.4.1, we restrict ourselves to considering a fixed number of candidate segments (7×7=49 in our experiments) on $\mathbf{V}$ for each $\mathbf{U}(s_{U_m}, l)$. In this case, minimizing $\mathcal{J}$ is a cyclic shortest path problem which can be solved using Dijkstra's algorithm. The nodes of the shortest path correspond to the selected $\mathbf{V}(s_{V_1}, l_{V_1}), \ldots, \mathbf{V}(s_{V_M}, l_{V_M})$, and the cost of the shortest path is the contribution to the similarity score, S_l (eq.(3)), at length l. Eqs.(3) and (4) are used to compute the similarity of shapes $\mathbf{U}$ and $\mathbf{V}$.

3.5. Normalized Procrustes Distances

It is interesting to consider whether the 'confidence' with which each segment match is chosen (out of the candidate matches available) carries useful information about the quality of the overall match. To investigate this idea, we normalize the PDs associated with a segment of $\mathbf{U}$ and its candidate matches on $\mathbf{V}$ by making the average PD equal to 1, and then apply HPM as normal. This means that when a selected match is much better than the other candidate matches, it will make a very small contribution to the similarity score – recall that a small score indicates high similarity. Using this approach, the similarity of two shapes (eq.(4)) will be a crude measure of the average confidence with which each segment match is selected, rather than an absolute measure of shape similarity. The surprisingly good results achieved using this confidence measure (Section 4) highlights the strong correlation between shape similarity



(a) (b) (c)

Figure 2. The segment matches found by HPM for the shapes in Fig. 1.

and the uniqueness of the optimal segment matches. We discuss this further in Sec. 6.

4. Experiments and Evaluations

The proposed algorithm was applied to the shapes in Figure 1. The matched segments are shown in Figure 2. Note that similar features are generally well matched at the scale at which they become prominent, whereas genuinely dissimilar features (*e.g.* the horses' tails) are never well aligned.

We now consider the performance of HPM on benchmark data sets. Four variants of the algorithm are considered:

- GPM: global Procrustes matching (Sec. 3.2)
- HPM-F: fast HPM (Sec. 3.4.1)
- HPM-DP: dynamic programming HPM (Sec. 3.4.2)
- HPM-Fn: fast HPM with normalized PDs (Sec. 3.5)

Shapes were described by N=100 points in all cases.

4.1. MPEG-7 Shapes

The "Bullseye Test" on the MPEG-7 shape data set[1] has been used extensively to assess the performance of shape retrieval algorithms. The data set is composed of 1400 binary images with a single shape in each image. There are 70 different classes and 20 observations in each class. Some of the shapes are shown in Figure 3. In the bullseye test, a shape is presented as a query and the top 40 matches are retrieved (from the entire data set – the test shape is not removed). The task is repeated for each shape and the number of correct matches (out of a maximum possible 20) are noted. A perfect performance results in $1400 \times 20 = 28000$ matches. Results are given as a percentage of this maximum score. Table 1 shows our results and, to the best of our knowledge, the best published scores. The algorithms described in [12, 4, 17, 9, 10] all achieved scores $<80\%$.

[1] http://www.cis.temple.edu/∼latecki/research.html#shape

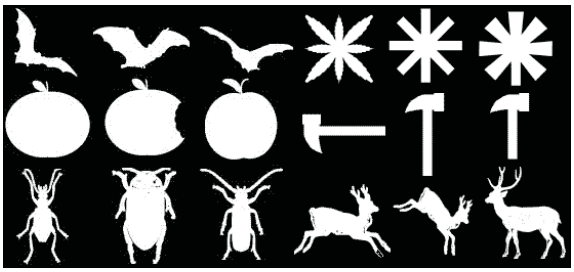


Figure 3. Example shapes from the MPEG-7 data set.

Table 1. Bullseye scores for best performing algorithms.

Algorithm	Score (%)
HPM-Fn	**86.35**
Internal Distances [13]	85.40
Multiscale Rep., Adamek *et al.* [1]	84.93
Polygonal Multiresolution [2]	84.33
HPM-F	**84.07**
HPM-DP	**83.98**
Chance Probability Functions [18]	82.69
Curvature Scale Space [16]	81.12
Generative Model [19]	80.03

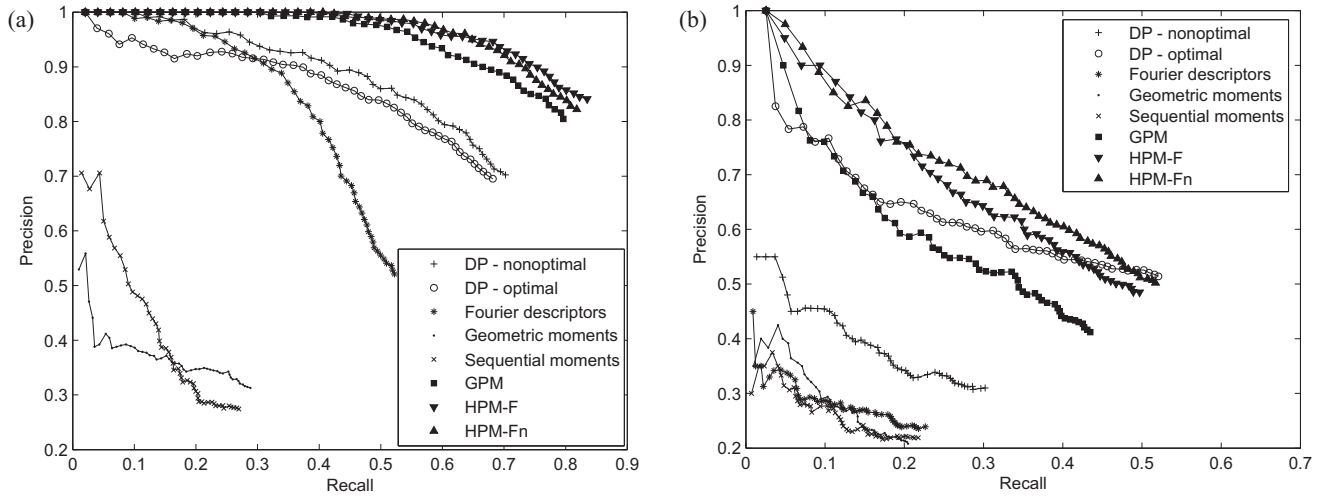


Figure 4. Precision-recall plots for (a) the GESTURES data, and (b) the MARINE data. Results for DP-nonoptimal, DP-optimal, Fourier descriptors, sequential moments, and geometric moments taken from Milios and Petrakis [15].

4.2. GESTURES Data

The data set consists of 980 shapes generated from 17 hand gestures. Milios and Petrakis [15] used the 17 original shapes as queries and human relevance data as the ground truth with respect to which different algorithms can be evaluated.[2] Figure 4a shows the precision-recall plots[3] from [15] augmented with our own results – the results for HPM-DP were very similar to those for HPM-F and have been omitted for clarity. Both GPM and HPM outperform the segment-based method described in [15]. The algorithms described in [3, 17, 2] have also been tested on this data.[4] From a visual inspection of their results, it seems that HPM outperforms Shape Contexts [3], performs similarly to the algorithm of Attalla and Siy [2] and performs worse than Sebastian *et al*.'s curve alignment algorithm [17].

4.3. MARINE Data

The data set contains the outlines of 1100 different marine species [16].[5] There are 20 query shapes and human relevance information is again used to evaluate performance [15]. Precision-recall plots are shown in Figure 4b (the results for HPM-DP are similar to those of HPM-F and are omitted). Again, HPM outperforms the technique described in [15] except when a large number of shapes are retrieved. Note that this time the difference in performance between GPM and HPM is more pronounced. The algorithm of Attalla and Siy [2] has also been tested on this data set. From a visual inspection of their results, it is clear that HPM performs significantly better on this data set.

[2]GESTURES data set and relevance information for GESTURES and MARINE data available at: http://www.ced.tuc.gr/∼petrakis

[3]Precision is the number of relevant shapes retrieved divided by the total number of retrieved shapes. Recall is the number of relevant shapes retrieved divided by the total number of relevant shapes in the data set.

[4]We requested the relevant results from the authors of [17] and [2] at rather short notice and did not receive them in time for publication.

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

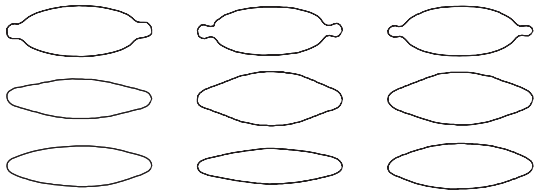


Figure 5. Example shapes from the DIATOM data set – shapes in the same row are in the same class.

4.4. DIATOM Data

The data set [10] consists of 781 outlines of individual diatoms.[6] The examples are not distributed equally among the 37 species, so Jalba *et al*. [10] applied a modified bullseye test whereby the number of retrieved shapes is set at twice the relevant class size. The 100 point polygonal approximations for some shapes from the data set are shown in Figure 5. We achieved bullseye scores of: GPM – 63.57%, HPM-F – 74.12% HPM-DP - 73.41% and HPM-Fn – 79.89%. HPM-Fn outperforms all but one of the techniques tested in [10]: Curvature Scale Space (CSS) [16] – 66%, Fourier descriptors – 73.7%, wavelet descriptors – 74.2%, Morphological Curvature Scale Space – 82.2%. Our results for this data set are likely to have suffered due to HPM's dependence on the global correspondence. The shapes have approximate rotational and reflectional symmetry (Figure 5) which suggests four potential choices of

[6]Diatoms are single celled algae with silica shells.

global correspondence between any two shapes. Ideally, all of these should be investigated using HPM.

4.5. Computational Complexity

The complexity of GPM scales linearly in the number of boundary points and hence, it is extremely fast ($<$10ms per match using 100 points). Matching two shapes takes 0.1-0.2s using HPM-F or HPM-Fn and $\sim$0.3s using HPM-DP on a 1.4GHz Pentium-M machine using interpreted Matlab code. It should be noted that the similarity score defined in eq.(4) requires two applications of HPM. The speed of HPM is similar to the algorithms in [19, 13], though slower than those described in [16, 18, 1, 2].

5. A Continuous Segment Mapping

It is easier to qualitatively assess the performance of HPM if one can examine the match between perceptual or functional elements of shapes – rather than the segments of pre-defined length and start point used by the algorithm. After applying HPM, we can use the finite number of matched segment pairs to find a continuous mapping which takes any segment of $\mathbf{U}$ to a segment of $\mathbf{V}$. Fig. 6 demonstrates this idea using thin plate spline regression.[7] It is clear from the figure that the mapping has matched the correct segments in both the cases shown. Note that the shorter segment of $\mathbf{U}$ (Fig. 6c) is a subsegment of the longer segment (Fig. 6b), yet the mapping shifts their start points in different directions and only changes the length of the shorter segment.

Rather than recovering a segment-segment mapping from HPM, one could formulate HPM in a continuous setting from the outset. Let us assume that $\mathbf{U}$ and $\mathbf{V}$ are parameterized closed curves with arc length parameters $s_U, s_V \in (0, 100]$, and use the same notation $\mathbf{U}(s_U, l_U)$ to denote a segment of $\mathbf{U}$. The aim of a continuous HPM algorithm would be to find a mapping $\mathbf{f}(s_U, l_U) = (s_V, l_V)$ such that $\mathbf{U}(s_U, l_U)$ corresponds to the segment $\mathbf{V}(s_V, l_V)$ of $\mathbf{V}$ (*c.f.* Fig. 6). Given a suitable family of functions $\mathcal{F}$, we could select the $\mathbf{f} \in \mathcal{F}$ which minimizes an appropriate cost functional. For example

$$\mathbf{f} = \arg\min_{\mathbf{f}\in\mathcal{F}} \int_{l=0}^{100} \int_{s=0}^{100} \{d(\mathbf{U}(s,l), \mathbf{V}(\mathbf{f}(s,l))) + \lambda \|\mathbf{f}''\|^2\} ds dl, \qquad (7)$$

where $d(\cdot,\cdot)$ is a measure of segment similarity and the final term is a smoothness penalty on $\mathbf{f}$ – segments of $\mathbf{U}$ with similar start points and lengths should be mapped to segments of $\mathbf{V}$ with similar start points and lengths. The ideas used to construct HPM-Fn in Sec. 3.4.2 were aimed at enforcing this 'smoothness' in a discrete setting. The minimum value of the functional in (7) would indicate the similarity of $\mathbf{U}$ and $\mathbf{V}$. There are additional ideas that could be considered when selecting $\mathcal{F}$ or designing the cost functional. For example, for a fixed l_U, s_V should be a monotonically increasing function of s_U so that the cyclic order of the segments of $\mathbf{U}$ is preserved on $\mathbf{V}$.

[7] Note that the regression gives the segment mapping. We are *not* using thin plate splines to deform one shape onto the other as in [5].

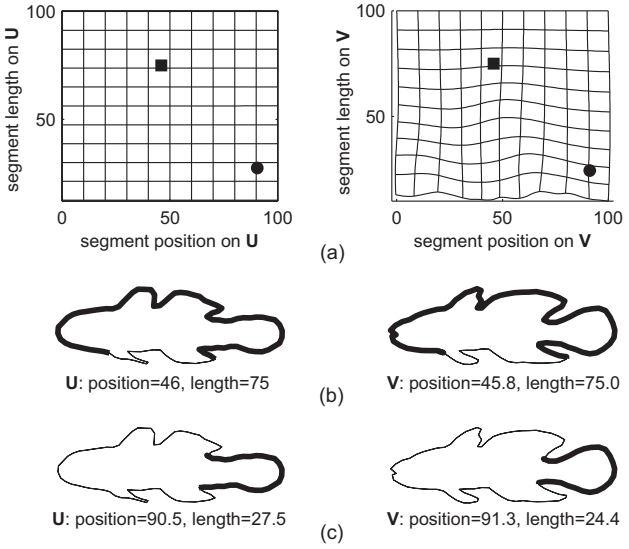


Figure 6. (a) Continuous segment mapping between two shapes. The highlighted points correspond to the segment matches shown in (b) – squares, and (c) – circles.

Formulating a continuous HPM in this way and applying continuous optimization techniques may provide a way of avoiding the large increase in computation associated with considering many candidate matches (*i.e.* using a larger or finer neighborhood grid). The soft approach of HPM-DP is likely to be required when many candidate matches are considered under the current framework, but finding the matches using dynamic programming is slow in such cases.

6. Summary and Discussion

We have introduced a novel shape matching algorithm which performs well on benchmark shape retrieval tests. Matching boundary segments of different lengths avoids problems associated with global and local approaches. The hierarchical structure of the matching algorithm captures the intuitive notion that matching should proceed in a global to local direction. While comparison of multiscale shape representations is typically based on specific features such as curvature-zero crossings (CSS [16]) or dominant points (wavelets [6]), with HPM there is no need to define such features. The proposed approach generalizes the idea of finding a point-to-point correspondence between two shapes to that of finding a segment-to-segment correspondence.

The experiments and results in Sec. 4 indicate that HPM is robust to a number of common transformations such as independent movement of parts and smooth deformations. Its reliance on segments makes it robust to viewpoint related changes in shape since the individual segments still match quite well even though the global alignment becomes poor (consider the bats class in Fig. 3). HPM requires ordered boundary information and assumes that the order in which corresponding segments appear is preserved across similar shapes. The ability of segments to stretch and slide along the boundary they are being matched to may enable HPM to handle missing features and slight occlusion, but it is not designed to handle significant occlusion or part rearrangement.

HPM-DP performed worst out of the different variants tested. A penalty-based approach to selecting the matches is likely to be more effective when segments are allowed to slide and stretch to a greater extent than was permitted here. It may only be necessary to use the predictions of the already matched longer segments in such cases and avoid the use of neighbor-based predictions which necessitate the use of dynamic programming. Also, the choice of penalty term has not been fully investigated – for example, the absolute difference between the predicted match and the chosen match (eq.(6)) may not be suitable.

The strong performance of HPM-Fn is particularly interesting. It seems likely that normalizing the PDs between a single segment of one shape and its candidate matches on the other shape leads to a simple form of novelty detection. For shape regions that have roughly constant curvature, all the candidate matches will be of a similar quality and all PDs will be close to 1. Consequently, the contribution of these regions to the similarity score will not be decisive. One could hypothesize that such regions are, on average, less useful for discrimination and that this effective down-weighting is desirable. On the other hand, there is often a definite best match for more complex segments (we can think of the segment clicking into place), and this will have a low normalized PD associated with it. Again, we might hypothesize that segments with an intricate structure (at the scale being considered) are more useful for discrimination and should have a greater impact on the similarity score. This is an interesting idea, though somewhat at odds with the notion of attaching equal significance to all parts of the shape and not identifying specific types of segment (Sec. 2). Future work will focus on investigating HPM-DP and HPM-Fn more thoroughly.

7. Acknowledgements

We wish to thank Euripides Petrakis, Farzin Mokhtarian, Longin Jan Latecki, and Andrei Jalba for making the data sets available.

References

[1] T. Adamek and N. O'Connor. A mutliscale representation method for nonrigid shapes with a single closed contour. *IEEE Trans. Circ. and Sys. for Vid. Tech.*, 14(5):742–753, 2004.

[2] E. Attalla and P. Siy. Robust shape simlilarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. *Patt. Recog.*, 38:2229–2241, 2005.

[3] S. Belongie and J. Malik. Matching with shape contexts. In *IEEE W'shop on Content-based Access of Im. and Vid. Libraries*, 2000.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24:509–522, 2002.

[5] F. L. Bookstein. Landmark methods for forms without landmarks: morphometrics of group differences in outline shape. *Med. Im. Anal.*, 1(3):225–243, 1996.

[6] L. F. Costa and R. M. Cesar. *Shape Analysis and Classification, Theory and Practice*. CRC Press, 2001.

[7] A. Del Bimbo and P. Pala. Shape indexing by multi-scale representation. *Im. and Vis. Comp.*, 17:245–261, 1999.

[8] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis*. Wiley, 1998.

[9] C. Grigorescu and N. Petkov. Distance sets for shape filters and shape recognition. *IEEE Trans. Image. Proc.*, 12(10):1274–1286, 2003.

[10] A. C. Jalba, M. H. F. Wilkinson, and J. B. T. M. Roerdink. Shape representation and recognition through morphological curvature scale spaces. *IEEE Trans. on Image Processing*, In Press.

[11] J. Latecki and R. Lakamper. Application of planar shape comparison to object retrieval in image databases. *Patt. Recog.*, 35:15–29, 2002.

[12] L. J. Latecki, R. Lakämper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages 424–429, 2000.

[13] H. Ling and D. Jacobs. Using the inner-distance for classification of ariculated shapes. In *CVPR*, 2005.

[14] G. McNeill and S. Vijayakumar. 2D shape classification and retrieval. In *IJCAI*, 2005.

[15] E. Milios and E. G. M. Petrakis. Shape retrieval based on dynamic programming. *IEEE Tran. Im. Proc.*, 1(1):141–147, 2000.

[16] F. Mokhtarian and M. Bober. *Curvature Scale Space Representation: Theory, Applications and MPEG-7 Standardization*. Kluwer Academic, 2003.

[17] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On aligning curves. *PAMI*, 25(1):116–125, 2003.

[18] B. J. Super. Learning chance probability functions for shape retrieval or classification. In *IEEE W'shop on Learning in Comp. Vis. and Pat. Recog.*, 2004.

[19] Z. Tu and A. Yuille. Shape matching and recognition using generative models and informative features. In *ECCV*, 2004.

[20] J. Zhang, X. Zhang, H. Krim, and G. G. Walter. Object representation and recognition in shape spaces. *Patt. Recog.*, 36:1143–1154, 2003.