11th IEEE International Conference on Computer Vision, Rio de Janeiro, 2007. ©IEEE.

Fast Matching of Planar Shapes in Sub-cubic Runtime *

Frank R. Schmidt
Computer Science Department
University of Bonn, Germany

Dirk Farin
University of Technology
Eindhoven, The Netherlands

Daniel Cremers
Computer Science Department
University of Bonn, Germany

Abstract

The matching of planar shapes can be cast as a problem of finding the shortest path through a graph spanned by the two shapes, where the nodes of the graph encode the local similarity of respective points on each contour. While this problem can be solved using Dynamic Time Warping, the complete search over the initial correspondence leads to cubic runtime in the number of sample points.

In this paper, we cast the shape matching problem as one of finding the shortest circular path on a torus. We propose an algorithm to determine this shortest cycle which has provably sub-cubic runtime. Numerical experiments demonstrate that the proposed algorithm provides faster shape matching than previous methods. As an application, we show that it allows to efficiently compute a clustering of a shape data base.

1. Introduction

The computation of distances between planar shapes is of fundamental importance in image analysis as it is a prerequisite for tasks such as shape clustering or the retrieval of similar shapes from a data base. The exponential increase of available visual data on the Internet creates a growing need for algorithms which compute such distances efficiently, as most of the above tasks require large numbers of pairwise shape comparisons. At the same time, many interesting shape metrics involve the computation of a *matching* which assigns a correspondence between points on either of the two shapes. In the present paper, we propose a new algorithm to solve this computational challenge efficiently.

The efficient computation of elastic matchings via dynamic programming techniques has a long tradition in the fields of string alignment, speech recognition, stereopsis and handwriting recognition [3, 10].

This approach has also been frequently applied to the matching of planar shapes [6, 2, 5, 9]. The key idea of the above approaches is to first select a corresponding pair of points on each of the two shapes — see the red circles in Figure 1. The subsequent matching of points on one shape to points on the other can then be solved by finding the shortest path through a graph, the nodes of which encode the similarity of respective point pairs. Local similarity can either be the difference of the curvature at points on either shape or some alternative integral invariants [9, 11]. If both shapes have n points, then the graph has n^2 nodes and the run-time of Dynamic Time Warping to find the shortest path is n^2. However, in order to consider all (non-self-occluding) matchings, one needs to consider all possible initial correspondences. In the above works, this aspect of the solution has been addressed by a complete search over all initial correspondences leading to an additional factor of n and thus cubic run-time. In [7] a cyclic string matching method was introduced and applied to shape matching [8, 12].

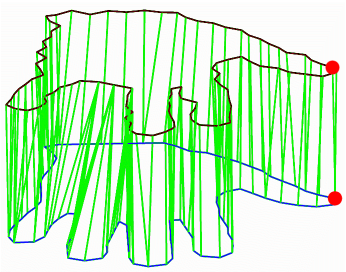

Figure 1. While the matching of two planar shapes (indicated by the green lines) can be computed efficiently using Dynamic Time Warping once an initial correspondence is known, most traditional methods require a complete search over the initial correspondence (red circles).

In this paper, we propose a more efficient matching algorithm by generalizing the fast algorithm for computing shortest circular paths on planar graphs [4] to graphs having a torus topology. In Section 2, we will develop this algorithm. In Section 3, we will prove that the runtime is at most $O(n^2 \log(n))$. Experiments demonstrate that runtimes compare favorably to those of alternative algorithms. As an application, we show in Section 4 that our algorithm allows to efficiently compute a clustering for a data base of shapes.

*This work was supported by the German Research Foundation, grant #CR-250/1.1

2. Matching of Planar Shapes

2.1. Shape Matching as a Shortest Cycle Problem

As a shape $\mathcal{C}$, we understand the class of closed curves $c : \mathbb{S}^1 \to \mathbb{R}^2$ that is invariant under rigid body motions. These shapes form the shape space $\mathcal{S}$ which can be described in means of the curves' curvature functions:

$$\mathcal{S} := \left\{ \kappa : \mathbb{S}^1 \to \mathbb{R} \,\middle|\, \kappa \text{ curvature of } c : \mathbb{S}^1 \to \mathbb{R}^2 \right\} \quad (1)$$

By introducing $\mathcal{S}$, all rigid body motions are eliminated and we can focus on the non-rigid shape transformations. To detect local transformations like stretching and contraction, we are looking for a correspondence mapping that maps the points of one shape to the corresponding points on the second shape. Since the points of a shape define an arbitrary subset of the plane $\mathbb{R}^2$, it is much simpler to find the correspondence directly on the parameterization domain $\mathbb{S}^1$. To ensure that a matching covers both parameterization domains exactly once, a matching consists of two orientation preserving bijective mappings $m_x, m_y : \mathbb{S}^1 \to \mathbb{S}^1$ that simultaneously sample points of both parameterization domains. Given two shapes $\mathcal{C}_1$ and $\mathcal{C}_2$ with their curvature functions κ_1 resp. κ_2, we are interested in a matching $m = (m_x, m_y)$ that minimizes the following functional

$$E^{\kappa_2}_{\kappa_1}(m) = \int_{\mathbb{S}^1} [(\kappa_1 \circ m_x - \kappa_2 \circ m_y)(s)]^2 \mathrm{dm}(s). \quad (2)$$

In this functional, the *data term* $(\kappa_1 - \kappa_2)^2$ is integrated along the *matching curve* $s \mapsto (m_1(s), m_2(s))$. Using (2), a distance function on the shape space $\mathcal{S}$ can be defined.

Definition 1 (Shape Distance)**.** Given two shapes $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{S}$ with their curvature functions $\kappa_1 : \mathbb{S}^1 \to \mathbb{R}$ and $\kappa_2 : \mathbb{S}^1 \to \mathbb{R}$ resp., we will call

$$\mathrm{dist}(\mathcal{C}_1, \mathcal{C}_2) := \min_{m \in \mathrm{Diff}^+(\mathbb{S}^1)^2} E^{\kappa_2}_{\kappa_1}(m) \quad (3)$$

the *distance* of these shapes. Every matching fulfilling this minimum will be called *a minimal matching* of $\mathcal{C}_1$ and $\mathcal{C}_2$.

The algorithm proposed in the following efficiently computes a global optimum for functionals defined on $\mathrm{Diff}^+(\mathbb{S}^1)$ in a discrete setting. The field of applications therefore reaches well beyond that of matching planar shapes. To find the optimal m, a shortest circular-path search on a constructed graph $G = (V, E)$ will be pursued. In this construction, the nodes $v_{i;j} \in V$ each represent a possible point-match $(c_1(i), c_2(j))$ between the two curves. At each point-match, there exist three different ways how to proceed on the two curves. These three ways are represented by three outgoing edges from each node.

1. One can proceed one step on both contours. This is represented by edges of the form $(v_{i;j}, v_{i\oplus 1;j\oplus 1})$, where $\oplus$ denotes addition modulo n.

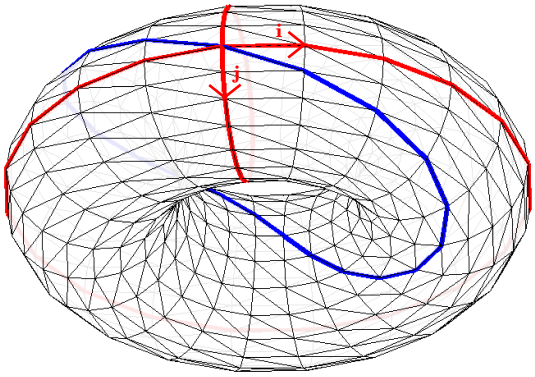

Figure 2. Shape matching as shortest path on a torus. Thin lines represent the graph edges. The thick blue line is one example of a matching between two shapes. Any feasible path traversing this torus induces a correspondence of points on one shape to points on the other.

2. One can proceed only on the first contour, which leads to edges of the form $(v_{i;j}, v_{i\oplus 1;j})$, or

3. only on the second contour, represented by edges $(v_{i;j}, v_{i;j\oplus 1})$.

The resulting graph G is illustrated in Figure 2. Matching both shapes now involves finding a closed path on G with minimum cost, where the cost is defined according to (2) in the following sense. Given two curvature vectors $k_1, k_2 \in \mathbb{R}^n$, we may define the pairwise dissimilarity $m_{i,j} := (k_{1,i} - k_{2,j})^2$ of $c_1(i)$ and $c_2(j)$. Now, we define the weight of an edge $e = (v_{i;j}, v_{k;\ell})$ by the average $\frac{m_{i,j}+m_{k,\ell}}{2}$ of their dissimilarity value. In doing so, the weight of every path is just the sum of the vertices' weights along the path.

2.2. Shape Matching with Planar Graphs

Computing the shortest circular path on G could be carried out with a brute-force method, where first an initial correspondence is chosen, which is used as a starting point for computing a best matching afterwards. This has to be repeated for all possible initial correspondences, leading to a computation time of $O(n^3)$. Instead, we propose to use the fast algorithm for shortest circular paths proposed in [4]. However, this algorithm is restricted to planar graphs, while our graph G has a cyclic, torus-like topology. In the following, we describe a modification of [4] such that the algorithm can be applied to the shape-matching problem.

First note that a shortest circular path on our torus-shaped graphs could also correspond to an inappropriate mapping, in which a contour is traced more than once, or not at all (by only selecting edges of types 2,3 as defined above). However, we are only interested in solutions in

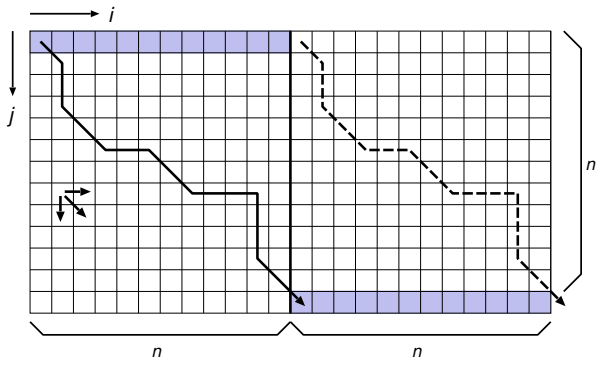

Figure 3. The path search is carried out on a graph of twice the size of the input matrix. A viable solution has to start in the top-left grey area and end at a matching node in the bottom-right grey area. The three allowed directions of the edges are indicated with the three arrows on the left side.

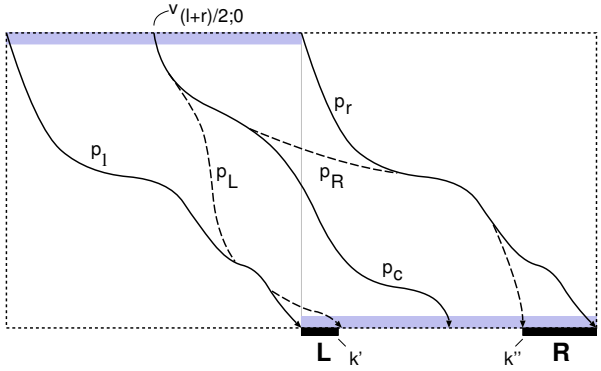

Figure 4. *(Step 2 and 3)* A shortest-path tree rooted at $v_c = v_{(l+r)/2;0}$ is computed. The right-most node $v_{k',n}$ for which $v_c \rightsquigarrow v_{k',n}$ still has a sub-path with v_l is determined. The shortest path to this node is denoted as p_L. Similarly, p_R is determined as the left-most path that still has a sub-path with p_r.

which each contour is traced exactly once. For this reason, we duplicate the cyclic graph G such that we obtain a graph of size $2n \times (n+1)$, defined as $G' = (V'; E')$ with $V' = \{v_{i;j} \mid 0 \leq i < 2n; 0 \leq j < n+1\}$ and edges $E' = \{(v_{i;j}, v_{i+1;j}), (v_{i;j}, v_{i;j+1}), (v_{i;j}, v_{i+1;j+1})\}$. This unfolded graph G' is planar and any path $v_{i;j} \rightsquigarrow v_{k;\ell}$ on G' corresponds to a path $v_{i \bmod n;j \bmod n} \rightsquigarrow v_{k \bmod n;\ell \bmod n}$ on the original graph G. The unfolding process allows us to explicitly count the number of wrap-arounds, since each wrap-around corresponds to an increase of n in the corresponding node coordinate.

Furthermore, when searching for the shortest circular path, we can restrict the search w.l.o.g. such that the start-node of the path is of the form $v_s = v_{i;0}$, $i = 0, \ldots, n-1$. The corresponding end-node for a single wrap-around in both coordinates is then $v_e = v_{i+n;n}$ (see Fig. 3).

2.3. Efficient Shape Matching

In the description of the algorithm for computing the shortest circular path, we will apply the following theorem:

Theorem 1. *Let $G = (V, E)$ with $V = \{v_i\}_i$ be a graph and let $p_1 = v_{i_1} v_{i_2} \ldots v_{i_n}$ and $p_2 = v_{k_1} v_{k_2} \ldots v_{k_m}$ be two minimum-cost paths. Then we can state that if p_1 and p_2 have two nodes v_p and v_q in common, there is a path $p_2' = v_{k_1} \ldots v_{k_m}$ with the same cost as p_2, which has a common sub-path $v_p \rightsquigarrow v_q$ with path p_1.*

A direct consequence of this theorem is that for any two minimum-cost paths p_1,p_2, there exist two paths p_1',p_2' with the same start and end nodes, which cross at most once. This property allows us to reduce the search area by constraining it on each side with a previously computed minimum-cost path.

The algorithm for computing the minimum cost path $v_{i;0} \rightsquigarrow v_{i+n;n}$ with unknown $i < n$ proceeds as follows.

- **Step 1:** The shortest path p_l from $v_{0;0}$ to $v_{n;n}$ is computed with a standard DTW algorithm. Furthermore, we define the path p_r as a copy of p_l, shifted by n elements in the i-direction, i.e., $v_{n;0} \rightsquigarrow v_{2n;n}$. The paths p_l and p_r are depicted in Figure 3 as the bold path and the dashed path, respectively. Note that these two paths constitute boundary paths which reduce the search area from $2n^2$ to at most $n^2 + n$ nodes.

- **Step 2:** This step makes use of previously defined left and right bounding paths $p_l = v_{l;0} \rightsquigarrow v_{l+n;n}$ and $p_r = v_{r;0} \rightsquigarrow v_{r+n;n}$. In the first iteration, the paths are taken from the result of Step 1, so that $l = 0$ and $r = n$. In later iterations, other bounding paths will be used.

  We now compute a shortest-path tree, starting from the middle node $v_c = v_{(l+r)/2;0}$ at the top side of the graph (Fig. 4). In one run of the DTW algorithm, we obtain all shortest paths to the nodes $v_{k;n}$ with $k \in l+n, \ldots, r+n$ at the bottom side. Note that we can limit the DTW computation to the area between the two bounding paths p_l, p_r.

- **Step 3:** If we consider the shortest paths between v_c and the bottom side of the graph, we see that the path $v_c \rightsquigarrow v_{l+n,n}$ obviously has a common sub-path with p_l, since both paths end at the same node. As we consider destination nodes $v_{k;n}$ with $k > l+n$, there is generally a largest k' with $l+n \leq k' \leq (l+r)/2+n$ so that the shortest path $v_c \rightsquigarrow v_{k';n}$ still has a common sub-path with p_l. Let us denote the path from v_c to $v_{k';n}$ as p_L, like it is depicted in Fig. 4.

  Similarly, we can find a smallest k'' with $(l+r)/2 + n \leq k'' \leq r+n$ so that the shortest path $v_c \rightsquigarrow v_{k'';n}$ still has a common sub-path with p_r. This defines the shortest path $p_R = v_c, \ldots, v_{k'',n}$.

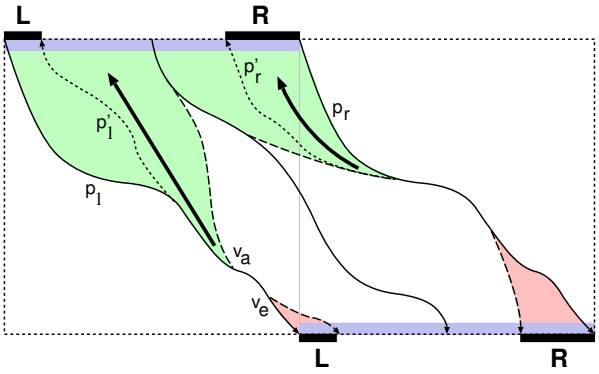

Figure 5. *(Step 4)* Given p_l and p_c, we know that all shortest circular paths ending in range $\mathbf{L}$ include the sub-path $v_a \rightsquigarrow v_e$. Hence, build a shortest-path tree, rooted at v_a, in the indicated direction on the shaded area. By combining these paths with the sub-path $v_a \rightsquigarrow v_e$ and the shortest-path tree below v_e, all circular paths through the range $\mathbf{L}$ can be obtained. The path p'_l is the shortest circular path $v_{k'-n;0} \rightsquigarrow v_{k';n}$, which will be used as a bounding path in the following recursion step.

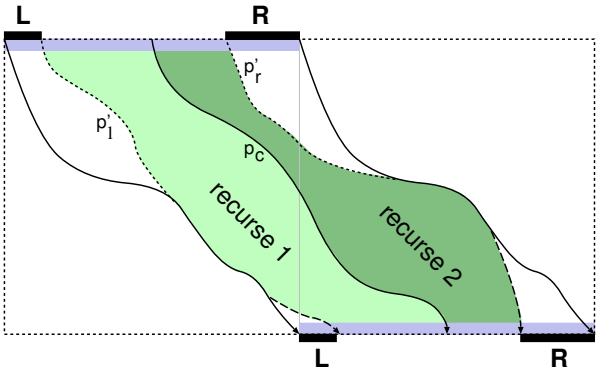

Figure 6. *(Step 5)* All circular paths through the ranges $\mathbf{L}$ and $\mathbf{R}$ are already computed. Only the shortest circular paths in the range in between remain unknown. This range is processed recursively, first processing the graph between p'_l and p_c, and then between p_c and p'_r.

- **Step 4:** Any circular path $v_{i;0} \rightsquigarrow v_{i+n;n}$ with $i \in \mathbf{L} = \{l+n, \ldots, k'\}$ is known to include the common sub-path of p_l and p_L. Let us denote the first node of this sub-path as v_a and the last node as v_e. We can now compute the shortest circular path that ends within the range $\mathbf{L}$ in a single step. To this end, we use DTW to compute a shortest-path tree, rooted at v_a, extending to the top-left and bounded by p_l and p_L (see Fig. 5). We now have the shortest-path tree rooted at v_a to all nodes $v_{i;0}$, $i+n \in \mathbf{L}$ and (still from the processing of Step 2) the shortest-path tree rooted at v_e to all nodes $v_{i;n}$, $i \in \mathbf{L}$. By considering the sum of the cumulative costs for pairs of nodes $v_{i;0}, v_{i+n;n}$, we can identify the shortest circular path for the node range $\mathbf{L}$. A similar process can be carried out to find the shortest circular path for the node range $\mathbf{R} = \{k''; \ldots, r+n\}$.

  Finally, we use the two shortest-path trees like above to extract the shortest-circular paths $p'_l = v_{k'-n,0} \rightsquigarrow v_{k',n}$ and $p'_r = v_{k''-n,0} \rightsquigarrow v_{k'',n}$. These two paths will constitute new bounding paths for later processing iterations.

- **Step 5:** The previous step has computed shortest circular paths for the ranges $l+n, \ldots, k'$ and $k'', \ldots, r+n$. What remains, is to compute the shortest circular paths for the range $k'+1, \ldots, k''-1$. Since we also know the circular path $p_c = v_c \rightsquigarrow v_{(l+r)/2+n;n}$, we can divide the computation into two subgraphs, bounded by the paths p'_l to p_c and p_c to p'_r. Both of these subgraphs can be processed recursively by restarting the processing at Step 2 for each of them. In the recursion, the new $p_l := p'_l$ and $p_r := p_c$ for the left subgraph, and

  $p_l := p_c$ and $p_r = p'_r$ for the right subgraph (Fig. 6).

Each processing of Step 4 gives up to two candidate shortest circular-paths. Once the whole range of start nodes is processed, the path with the minimum-cost path is selected as the global solution.

3. Runtime Analysis

In this section, we analyze the proposed shape matching method and compare it to state of the art methods. First, we will prove that the worst case complexity of our method is $O(n^2 \log(n))$ where n is the number of sample points on both shapes. In the second subsection, a direct runtime comparison will be provided.

3.1. Sub-cubic Runtime

The proposed method of Section 2.3 defines a planar graph $G = (V, E)$ on a rectangular grid $V' = \{v_{i;j} \mid 0 \leq i < 2n; 0 \leq j < n+1\}$. We are now looking for a shortest path from $v_{i;0}$ to $v_{i+n;n}$ where i varies over the set $\{0, \ldots, n-1\}$. Since our method works recursively, we are especially interested in any connected subgraph (V', E') of G that includes a certain subset of the two boundaries $B_1 := \{v_{0;0}, \ldots, v_{n-1;0}\}$ and $B_2 := \{v_{n;n}, \ldots, v_{2n-1;n}\}$.

Lemma 1. *Let $G' = (V', E')$ be a connected subgraph of G containing $b > 1$ corresponding boundary elements, i.e.*

$$v_{i;j} \in (V' \cap B_1) \Leftrightarrow v_{i+n;j+n} \in (V' \cap B_2),$$

where $b = |V' \cap B_i|$ for $i = 1, 2$. Then the algorithm finds the shortest path connecting an element $v_{i;j} \in B_1$ to its corresponding element $v_{i+n;j+n} \in B_2$. Moreover, the number of calulation steps $T(N, b)$ is bounded from above by $N(\log(b-1)+2) + 2n(b-1)\log(b-1)$ where $N := |V'|$ is the number of vertices in G'.

Proof. Since G' is a subgraph of G, any shortest path given a start vertex and a target vertex can be calculated within N calculation steps using Dynamic Time Warping (DTW). This property will be used during this proof. We like to prove this lemma by complete induction over the boundary length b.

Initialization: For $b = 2$, the problem can be solved by two DTW runs. Therefore, the upper bound holds.

Induction step: Assuming, we have proven the upper bound for all boundary lengths $1 < b' < b$. Now, we like to prove this upper bound for b itself. First, the algorithm calculates a shortest path from the central point of the boundary which takes N calculation steps. By doing so, the graph G' is split into a left and a right subgraph with N_L resp. N_R vertices whereas $N_L + N_R \leq N + 2n$. If in any of these subgraphs the two boundaries touches one another, we can calculate a valid minimal path in the subgraph in N_L resp. N_R steps. Otherwise, the induction hypothesis for $b' < \frac{b+1}{2}$ is applicable and the runtime is bounded from above by

$$\begin{aligned} & N + T\left(N_L, \frac{b+1}{2}\right) + T\left(N_R, \frac{b+1}{2}\right) \\ & \leq N(\log(b-1)+2) + 2n\left(b\log\frac{b-1}{2} + 2\right) \\ & \leq N(\log(b-1)+2) + 2n(b-1)\log(b-1) \end{aligned}$$

□

With this lemma, the worst case runtime of the proposed matching algorithm can be characterized as follows.

Theorem 2. *The proposed shape matching algorithm has a worst-case complexity of* $O(n^2 \log(n))$.

Proof. Since $G = (V, E)$ is a subgraph of itself with $b = n$ boundary elements and $N = n^2 + n$ vertices, Lemma 1 guarantees that the shortest path can be calculated in less than $3n^2 \log(n-1) + 2n^2 - n\log(n-1) + 2n$ steps. □

3.2. Experimental Comparison

The above bound of $O(n^2 \log(n))$ on the computation time was proven based on a worst case analysis. In practice, the computation time is well below this worst case bound: As discussed in Step 4 of Section 2.3, the algorithm cuts away parts of the graph, for which it can immediately compute the optimal solution. For most shape comparisons, such cases arise instantly, such that the recursion terminates after very few iterations.

Figure 7 shows a quantitative benchmark test of the proposed method and three state-of-the-art shape matching algorithms:

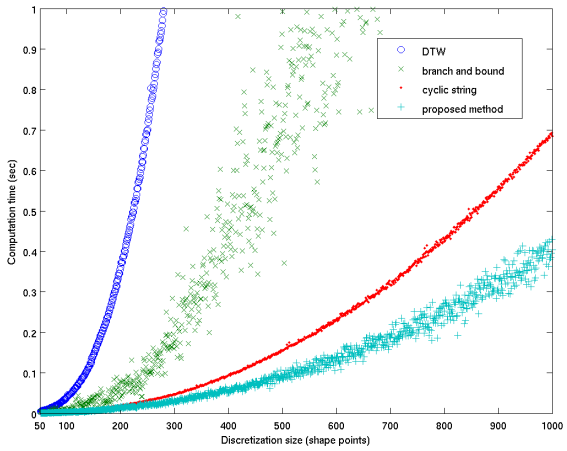

Figure 7. Experimental runtime comparison. In contrast to DTW, Branch-and-Bound and the cyclic-string approach [8], the proposed recursing cyclic matching algorithm exhibits consistently lower run-times, in particular for larger problem size.

Dynamic Time Warping For every possible initial match $(i, 0)$, we look for the shortest path from the point $(i, 0)$ to the point $(i + n, n)$ within the graph $G = (V, E)$ using dynamic programming (DTW). To find a match, we always need $O(n^3)$ calculation steps.

Branch and Bound This method divides the set of initial matching recursively in subsets $S \subset \{(0, 0), \ldots, (0, n-1)\}$. Then the DTW method looks for a shortest path through S until the shortest path is a valid matching, i.e. a cycle in the graph $G = (V, E)$. The worst case is still $O(n^3)$, but under some conditions, an average case of $O(n^2 \log(n))$ is possible [1].

Cyclic String Approach In [8], a shape matching approach was presented that essentially uses Step 1, 2 and 5 of our approach.

For two shapes and an increasing discretization level which ranges between 50 and 1000 points per shape, we compared the runtime of the described methods. While all algorithms compute the same matching, the proposed method exhibits consistently lower run-times, in particular for larger discretization. Moreover, it offers a predictable performance, in the sense that the computation times exhibit a smaller spread than those of Branch and Bound.

4. Application: Shape Clustering

Minimizing the functional (2) not only provides a matching between two given shapes. The cost of the shortest path can be interpreted as a distance between the two shapes.

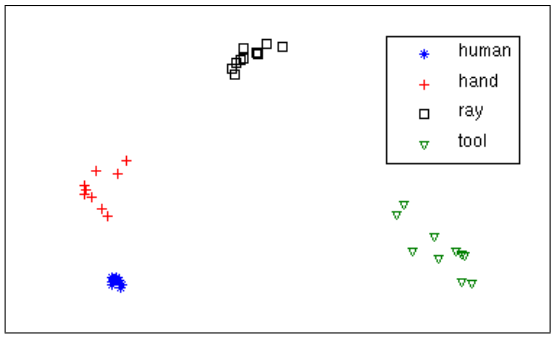

Figure 8. Shape Clustering. For a database of 40 shapes from four different classes, we computed the pairwise distances (3) using the algorithm presented above. Based on the matrix of pairwise distances, we computed a clustering and a $2D$-embedding of all shapes. The plot shows that the clustering respects the semantic of the given shapes. This indicates that the proposed matching algorithm provides shape distances which reproduce the human notion of shape similarity for this data base.

In order to demonstrate that the shape distance (3) indeed captures a meaningful notion of shape dissimilarity, we applied the shape matching algorithm introduced above to compute a clustering of a set of $n = 40$ shapes $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$. To this end, we applied a curvature descriptor [11][1] on a subset of the LEMS shape database [13] which describes the shape classes *hand*, *human*, *ray* and *tool*. Finally, we computed the matrix D of pairwise distances

$$D_{ij}^2 = \mathrm{dist}(\mathcal{C}_i, \mathcal{C}_j), \quad \forall\, i, j = 1, \ldots, n. \tag{4}$$

and performed a k-means clustering with $k = 4$ clusters.

For visualization, we embedded all shapes into a two-dimensional space by multi-dimensional scaling (MDS), thereby optimally preserving the pairwise distances, i.e. we computed a set of points

$$\left\{x_1, \ldots, x_n \in \mathbb{R}^2, \text{ such that } |x_i - x_j|^2 \approx D_{ij}^2 \ \forall i, j\right\}.$$

Figure 8 shows these $2D$ points with their cluster membership color-coded. The clear separation of four clusters associated with the four shape classes indicates that the computed pairwise distances reproduce the human notion of shape similarity for this data base.

5. Conclusion

We proposed an efficient algorithm to solve the combinatorial problem of matching two planar shapes. The mapping of points along one contour to corresponding points along the other is determined by finding the shortest circular path on a torus, the nodes of which encode the local similarity of a robust curvature measure at respective point pairs.

[1]The curvature descriptor introduced in [11] differs from that in [9] in the sense that it is a *consistent* first-order approximation of the curvature.

In particular, our algorithm efficiently incorporates the search over the initial correspondence. The key idea is to slice the torus and convert it to a planar graph without reducing the solution of permissible matchings. Subsequently, we use a divide-and-conquer technique to cut down the search-space in the recursion steps. To the best of our knowledge, this constitutes the fastest algorithm for combinatorial matching of planar shapes published so far.

In a benchmark test, we demonstrated that the proposed algorithm consistently outperforms state-of-the art matching algorithms, providing problem-specific speed-up factors of 2 over the cyclic-string method [7] and up to 100 over the DTW method. In a shape clustering application, we showed that the computed shape distances reproduce human notions of shape similarity.

References

[1] B. C. Appleton. *Globally minimal contours and surfaces for image segmentation*. PhD thesis, University of Queensland, Australia, December 2004.

[2] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:2365–2385, 1998.

[3] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[4] D. Farin and P. H. N. With. Shortest circular paths on planar graphs. In 27^{th} *Symposium on Information Theory in the Benelux*, pages 117–124, June 2006.

[5] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classication of silhouettes. *IEEE PAMI*, 21(12):1312–1328, 1999.

[6] D. Geiger, A. Gupta, L. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking and matching deformable contours. *IEEE PAMI*, 17(3):294–302, 1995.

[7] M. Maes. On a cyclic string-to-string correction problem. *Inf. Process. Lett.*, 35(2):73–78, 1990.

[8] M. Maes. Polygonal shape recognition using string-matching techniques. *Pattern Recognition*, 24(5):433–440, 1991.

[9] S. Manay, D. Cremers, B.-W. Hong, A. Yezzi, and S. Soatto. Integral invariants for shape matching. *IEEE PAMI*, 28(10):1602–1618, 2006.

[10] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.

[11] F. R. Schmidt, E. Töppe, D. Cremers, and Y. Boykov. Efficient shape matching via graph cuts. In *EMMCVPR*, volume 4679 of *LNCS*, pages 39–54. Springer, August 2007.

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

[13] D. Sharvit, J. Chan, H. Tek, and B. Kimia. Symmetry-based indexing of image databases, 1998.