Probabilistic matching of sets of polygonal curves*

Helmut Alt† Ludmila Scharf † Sven Scholz †

1 Introduction

Analysis and comparison of geometric shapes are of importance in various application areas within computer science, e.g., pattern recognition and computer vision. The general situation in a shape matching problem is that we are given two shapes A and B and a certain class T of allowable transformations and we want to transform B optimally so that the transformed image of B is as close to A as possible. We assume that shapes are modeled by sets of polygonal curves. As possible class of transformations we will consider *translations*, *homotheties* (scaling and translation), *rigid motions* (rotation and translation), *similarities* (rotation, scaling and translation), and *affine maps*.

We address the problem of optimal matching of the complete shape B to the complete shape A, called *complete-complete matching*. The algorithm we present also applies to the problem of *complete-partial matching*, i.e., matching B completely as good as possible to some part of A, and *partial-partial matching*, i.e., matching some part of B as good as possible to some part of A.

Several similarity measures and algorithms are known to match two curves, especially polygonal curves, see [10] for a survey. One of the "universal" similarity measures is the Hausdorff distance which is defined for any two compact sets A and B. In [1, 3] Alt et al. describe efficient algorithms for computing the Hausdorff distance and minimizing it under translations and rigid motions for arbitrary sets of line segments. One of the drawbacks of the Hausdorff-distance is that it is very sensitive to noise. A few similarity measures are defined for pairs of curves, which capture the relative course of two curves: Fréchet distance [3], turning function distance [4], and dynamic time warping distance [6]. There are no generalizations of those distances to sets of curves, although in [2] a generalization of the Fréchet distance to geometric graphs is given, and in [9] Tanase et al. describe an algorithm for matching a set of polygonal curves to a single polygon. A similarity measure which is designed for sets of curves is the reflection visibility distance [7]. The reflection visibility distance is robust against different kinds of disturbances but is expensive to compute. Some of these similarity measures can be modified to valuate partial match, e.g., percentile-based Hausdorff distance [8].

The method we introduce is close to an intuitive notion of "matching", i.e., find one or more candidates for the best transformations, that when applied to the shape B map the most similar parts of the two shapes to each other.

2 Probabilistic matching

For given two shapes $A, B \subset \mathbb{R}^2$ represented by sets of polygonal curves we want to find a transformation t, which lets the transformed image of B, $t(B)$, *match best* A, i.e., maps the most similar parts of the shapes A and B to each other.

2.1 Simple matching algorithm

The idea of the *probabilistic approach* is quite simple:

1. Take small random samples S_A from A and S_B from B and give one "vote" to the transformation t which maps S_B to S_A.

2. Repeat this experiment many times. Then the distribution of votes in the transformation space T approximates a certain probability distribution.

3. For a given neighborhood size δ take the points of T with the highest number of votes in their δ-neighborhood as candidates for good transformations.

The idea behind this algorithm, is that the transformations, which map large parts of shapes to each other should get significantly more votes than others. The size of the δ-neighborhood influences the quality of the match.

The size of a random sample within one experiment depends on the class of transformations allowed:
For *translations*, S_A and S_B contain each a single randomly selected point of a correspondent shape. The transformation space is two-dimensional and the resulting translation is a vector in $\mathbb{R}^2$.

In case of *rigid motions* the transformation space is three-dimensional. A random sample of a shape within one experiment contains a random point and a

*This research was supported by the European Union under contract No. IST-511572-2, Project PROFI.

†Institute of Computer Science, Freie Universität Berlin

unit length vector corresponding to the average direction of tangent lines of its neighborhood. Two such point-vector pairs define uniquely a rigid motion.

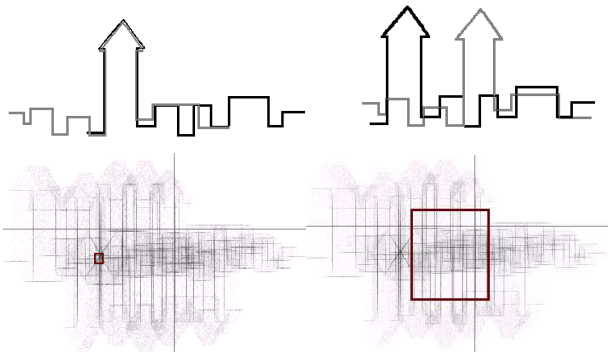
For *similarity maps* the transformation space is four-dimensional and a random sample from a shape contains two points. Two pairs of points a_1, a_2 in A and b_1, b_2 in B determine a unique four-dimensional similarity transformation t mapping b_1 to a_1 and b_2 to a_2.

In general, we define the δ-neighborhood of a transformation t as a set of transformations that map any point b in B into a δ-neighborhood of the point $t(b)$.

2.2 Analysis for the Translations

As a detailed analysis shows, for most cases the translation with most votes brings the largest fitting parts of A and B into the δ-neighborhood of each other.

The choice of δ therefore controls the trade-off between the quality of match and the size of the parts matched. With a small value of δ our algorithm would find a translation which maps nearly congruent parts of two shapes to each other, see Figure 1(a). A large value of δ leads to a translation which gives a rough match but for larger parts of the shapes, see Figure 1(b).



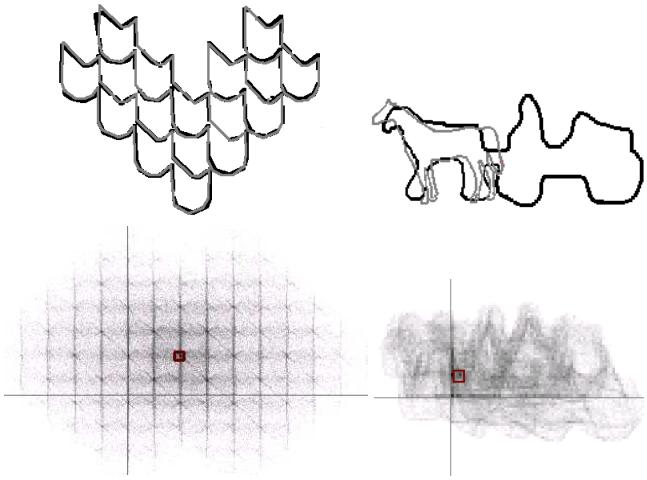
(a) (b)

Figure 1: Matching with (a) small grid size and (b) large grid size

But the value of δ does not alone determine what kind of matching we get or how large the matched parts are. For nearly congruent figures a small neighborhood size already leads to a complete-complete matching, see Figure 2(a), and with the same value for δ we can get complete-partial matching, see Figure 2(b).

Furthermore, we should examine several local maxima of the vote distribution function, since each of the maxima corresponds to a partial match between two figures, an example is illustrated in Figure 3.

For most applications it would make sense to determine for each of the candidate transformations which parts of the two shapes match and how large these



(a) (b)

Figure 2: Matching with a sampling neighborhood of the the same size in translation space for different shapes

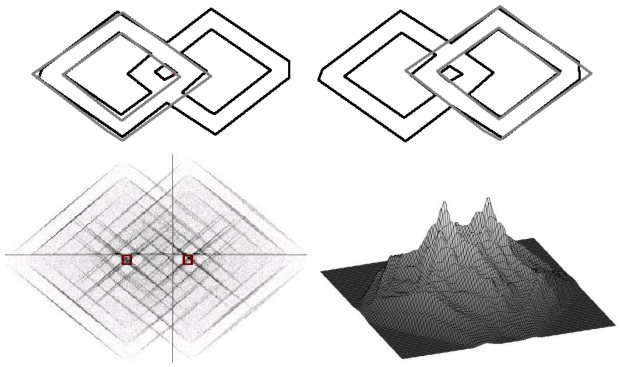


Figure 3: Top: two superimpositions of the figures each corresponding to a complete-partial match. Bottom: Experimental distribution in the translation space with two local maxima marked and a 3d-view of the distribution.

parts are. This verification step can be performed using the directed Hausdorff distance. We can incrementally take the segments of the shape B into the matched set if the directed Hausdorff distance from the matched subset of B to A stays under the chosen value δ. With the algorithm from [1] we can perform this verification step in time $O((n+m)\log(n+m))$, where n and m are the numbers of segments in A and B, respectively.

The problem of partial-partial matching is not uniquely defined since there is a certain correlation between the quality of match and the size of the matched parts. We address this problem by letting the user specify the quality of match through the choice of δ, for which we then find the matching parts.

Running time. Maximizing the the number of votes in the δ-neighborhood of a translation, we maximize the measure of the set $M(t) = \{(a,b) : a \in A, b \in B, \|a - t(b)\| \leq \delta\}$. We define a similarity measure associated with a translation t as $f(t) = |M(t)| / |A \times B|$. Let $\tilde{f}(t)$ denote the ratio of the number of sample translation vectors in the δ-neighborhood of t to the total number of samples. $\tilde{f}(t)$ is the estimate of $f(t)$. Using the technique described in [5] we can bound the absolute error for the estimate of the distribution of votes in a following way:

The number of sample translation vectors sufficient to get an approximation error of at most ε with probability at least $1 - \eta$ is $N = \frac{16 \ln \frac{1}{\eta}}{\varepsilon^2} + 2$. With linear time preprocessing we can generate a random point of a shape modeled by n line segments in time $O(\log n)$. That is, the time to generate N random points on both shapes is $O(n + N \log n)$.

In order to find a translation with the highest number of votes, or the sampling points, in its δ-neighborhood, we consider the arrangement of the δ-neighborhoods of the sampling points. The basic observation is that if a sampling point s is contained in the δ-neighborhood of the translation t, then t is also contained in the δ-neighborhood of s. All translations in the same cell of the arrangement have the same sampling points in their δ-neighborhoods. Therefore it is sufficient to traverse the arrangement and take the nodes with the highest number of sampling points whose δ-neighborhoods contain this node. The arrangement of the δ-neighborhoods of N vectors has the complexity $O(N^2)$ and can be computed and traversed in time $O(N^2)$, which yields the following theorem:

Theorem 1 *Given two shapes modeled by sets of line segments of complexity n in the plane and parameter ε, η, $0 \leq \varepsilon, \eta \leq 1$. In time $O(n + \frac{\log \frac{1}{\eta} \log n}{\varepsilon^2} + \frac{\log^2 \frac{1}{\eta}}{\varepsilon^4})$ we can compute a translation t_{app} such that $\left|\tilde{f}(t_{\text{app}}) - f(t_{\text{opt}})\right| \leq \varepsilon$ with probability at least $1 - \eta$, where t_{opt} is a translation maximizing $f(t)$.*

2.3 A Faster Heuristic

We are also working on heuristics, in order to speed up the matching process, that is to find good transformation candidates with less experiments. Here we describe an algorithm which uses a set of pairs of corresponding points and during one random experiment iteratively extends that set until no further data are available or the samples are no longer consistent. For every set a transformation is computed and the best one is registered and gets a weight. Then we get a weighted sample of the transformation space, where the neighborhoods with large weight are likely to contain candidates for transformations resulting in a good match for the shapes.

The problem of computing preliminary transformations consists of two subproblems: one is to find correspondences between features, the other is to find a transformation that maps the corresponding features to each other.

Finding Correspondences. The initial part of every vote is the random choice of a single vertex of each of the two sets of polylines, and the direction for traversing the list of following vertices (also both possible directions may be processed). Starting from that pair a sequence of sets of pairs of vertices and vertex surrogates is generated.

Let p_0 be the randomly chosen vertex from the first set S_1 of planar polylines and let $p_1, \ldots, p_k$ be the succeeding vertices with respect to the randomly chosen direction. Analogously, let q_0 be the randomly chosen vertex from the second set S_2 of planar polylines and let $q_1, \ldots, q_l$ be the succeeding vertices. The pair (p_0, q_0) is added to the – so far empty – sample set s.

In each iteration step distances from the last added pair of points to the next vertices on the corresponding polylines are computed. If the two computed distances are nearly equal, the next two vertices are taken as a corresponding pair which is added to the sample set s.

Otherwise a vertex surrogate is created for the polyline with a larger distance. A surrogate is a point lying on an edge of the polyline, but nevertheless is treated like a vertex. It is chosen to have the same distance to its predecessor as the corresponding two vertices of the other polyline have.

When the end of a polyline is reached, then starting from the initial pair the traversal is performed in the opposite direction.

Calculating the Transformations. For every new pair of vertices or vertex surrogates added to s the transformation $t \in T$ is computed that minimizes the sum of the weighted squared distances $\varepsilon(t) = \sum_{(p_i,q_i) \in s} w(p_i, q_i) \|q_i - t(p_i)\|^2$ with $w(p_i, q_i)$ being half the length of the edges incident to p_i and q_j.

For T being the class of translations or rigid motions, t and the vertex correspondences may be determined independently from each other. But for the classes of transformations that allow scalings (i.e. homotheties, similarities, and affine maps), the assignment of the vertices depends on that scaling.

In every iteration step it is checked whether the error introduced by the new added pair is still within tolerance bounds. If the error is too big, the traversal of the polylines is ceased, as illustrated in Figure 4: the bold polylines are traversed up to the end of the dashed parts. The transformation for which the error was farthest from the tolerance bound is weighted and

handed over to the clustering algorithm (the transformation calculated for the bold polylines up to the beginning of the dashed line in the example).

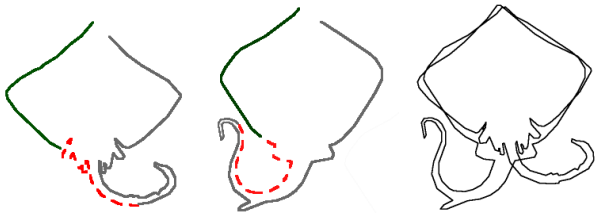


Figure 4: Two instances of the mpeg7shapeB dataset (ray-7, ray-20 and both mapped).

Scalings: If scalings are allowed, for every single vote a prescaling factor c_i is randomly chosen such that $\mathrm{ld}(c_i)$ is normally distributed with mean value $\mathrm{ld}(\bar{c}_i)$ where $\bar{c}_i$ is the prescaling factor of the transformation rated best so far. $S_1^{c_i}$ denotes the set S_1 scaled by c_i. For the rest of the vote, every operation concerning the first set S_1 (e.g. computing the distance between two vertices) is performed on $S_1^{c_i}$.

Weighting the Transformations and Clustering. The two factors that have to be considered for weighting a transformation t are the expressiveness of the sample and the quality of the match. Let ε be the sum of weighted squared distances, let $w(s)$ be the sum of all weights of the pairs of points in the sample set s, and let d_{bb} be the diameter of the bounding-box containing the covered part of the polyline. Defining the relative root mean square error $e = \sqrt{\varepsilon/w(s)}/d_{bb}$ yields a value representing the quality of the match which is invariant under scalings.

The match score or weight $w(t)$ of a transformation t is then defined as $w(t) = l/(1+\gamma \cdot e)$ with γ being an arbitrarily chosen constant for balancing out the impact of the length l and the error e.

Let t_1 and t_2 be two arbitrary transformations and let S be the transformed shape. The distance measure $d_S(t_1, t_2) = \max_{p \in S'} \|t_1(p) - t_2(p)\|$, with S' being the set of the vertices of the bounding box of S forms a metric space for affine maps, under the assumption that the four points of S' are pairwise different.

A cluster in our sense represents a region of limited diameter, which subsumes a considerable amount of weights of the enclosed input points (transformations).

Let T_n be the set of n preliminary transformations and w_i be the weight of transformation $t_i \in T_n$. For every transformation t_k the set of its dominators in range r is defined as a set of transformations with distance at most r to t_k with a weight greater than w_k, which have no dominators in range r. Every transformation t_c which has no dominators in range r_c defines a cluster with radius r_c as the set $\{t_i \in T_n | d(t_c, t_i) < r_c\}$, i.e., a transformation is either assigned to its dominators' clusters or it defines a cluster itself. The weight of a cluster is defined as the sum of the weights of its elements. This definition allows for a fast computation of all clusters and their weights.

3 Conclusion

We presented a probabilistic approach for matching two shapes represented by sets of polygonal curves, which is close to the human notion of match and is easy to implement. The algorithms are also applicable to the problem of partial matching and we obtained convincing results from experiments with the MPEG7 shape dataset.

References

[1] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13:251–265, 1995.

[2] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *J. of Algorithms*, pages 262–283, 2003.

[3] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of computational geometry*, pages 121 – 153. Elsevier Science Publishers B.V. North-Holland, 1999.

[4] E. Arkin, P. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–215, March 1991.

[5] O. Cheong, A. Efrat, and S. Har-Peled. On finding a guard that sees most and a shop that sells most. In *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 1091–1100, 2004.

[6] A. Efrat and S. Venkatasubramanian. Curve matching, time warping, and light fields. Technical Report AT&T TD-4z5TMU, AT&T.

[7] M. Hagedoorn, M. Overmars, and R. Veltkamp. A new visibility partition for affine pattern matching. In *Proc. Discrete Geometry for Computer Imagery conference, DGCI 2000*, pages 358–370, Berlin, 2000. Springer-Verlag.

[8] M. Hagedoorn and R. Veltkamp. State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.

[9] M. Tanase, R. C. Veltkamp, and H. Haverkort. Multiple polyline to polygon matching. In *Proceedings of the 16th Annual Symposium on Algorithms and Computation (ISAAC 2005), LNCS 3827*, pages 60–70, 2005.

[10] R. C. Veltkamp. Shape matching: Similarity measures and algorithms. Technical Report UU-CS-2001-03, Utrecht University, 2001.