Probabilistic Matching and Resemblance Evaluation of Sets of Polygonal Curves ⋆

Ludmila Scharf, Sven Scholz

Institute of Computer Science, Freie Universität Berlin

Abstract

This paper presents a novel matching and similarity evaluation method for planar geometric shapes represented by sets of polygonal curves. Given two shapes, the matching algorithm randomly generates a point sample from each shape and records a vote for a transformation which maps one sample to the other. The experiment is repeated many times. Clusters of votes in the transformation space indicate good candidate transformations for matching the two shapes. Unlike most voting schemes, though, the samples taken in one random experiment are extended as much as possible and the vote is weighted depending on the samples. The best clusters are those with a large total weight. The second part of the method is a resemblance evaluation of the two matched shapes. We give definitions of resemblance functions for complete and for partial matching. The system is evaluated using the MPEG-7 shape silhouette database and a collection of 10 745 trade mark images. The experiments demonstrate a high performance of our algorithms for contour shapes as well as for trademark images.

Key words: Trademark image retrieval, Shape matching, Probabilistic algorithms, Shape similarity

1 Introduction

Motivated by the task of automated retrieval of figurative images we developed an algorithm for the evaluation of shape similarity. It consists of two main phases: matching and evaluation of the resemblance of the matched shapes.

⋆ This research was supported by the European Union under contract No. IST-511572-2, Project Perceptually-Relevant Retrieval of Figurative Images (PROFI).

Email addresses: scharf@inf.fu-berlin.de (Ludmila Scharf), scholz@inf.fu-berlin.de (Sven Scholz).

The approach we introduce for matching two geometric shapes S_1 and S_2 modeled by sets of plane polygonal curves, is close to an intuitive notion of "matching", i.e., find one or more candidates for the best transformation, that when applied to the shape S_1 maps the most similar parts of the two shapes to each other. As allowable classes of transformations we will consider *translations*, *homotheties* (scaling and translation), *rigid motions* (rotation and translation), *similarities* (rotation, scaling, and translation), and general *affine maps*.

The matching step of our algorithm is a voting scheme. Unlike in most well known approaches including geometric hashing [1], pose clustering (generalized Hough transform) [4–7], and the random sample consensus (RANSAC) [3], we do not use a minimum sample of features to compute the model parameters (the matching transformations in this case), but we find large consistent sequences of the corresponding points of two shapes voting for the same transformation. This transformation gets a vote which is weighted depending on the size of the sample and the quality of the match. Thus, we get a distribution of weighted votes in the transformation parameter space. Similar as in the pose clustering approach, we then take the largest clusters as candidate transformations, where largest clusters are those with the highest total weight.

After several candidate transformations of one shape have been identified by the matching algorithm, each of these transformations t_i is applied to the shape S_1 and the similarity of the shape S_2 and the transformed shape $t_i(S_1)$ is validated using the resemblance function described in section 3. The proposed resemblance function incorporates two perceptual factors: proximity and parallelism (or factor of direction), that is, the resemblance value is high if the distances between the points of two shapes are small and the line segments contained in the shapes are nearly parallel. The transformation with the highest similarity value is then selected as the best match.

We address the problem of matching the complete shape S_1 to the complete shape S_2, called *complete-complete matching* (CCM). In addition, we consider the problem of *complete-partial matching* (CPM), i.e., matching S_1 completely as good as possible to some part of S_2, and *partial-partial matching* (PPM), i.e., matching some part of S_1 as good as possible to some part of S_2. Clearly, both partial matching problems CPM and PPM are not uniquely specified since there is a trade-off between the quality of the match and the size of the matched parts. Which of these two criteria is more important, depends on the application. We address this problem by introducing a parameter which regulates the influence of the quality of match and the matched size on the similarity value.

Both, the matching procedure and the resemblance function are designed to be robust with respect to noise and to differences in the representations of the

the shapes. The data used in the experiments described in section 4 show the importance of that robustness.

2 The matching algorithm

Given two sets S_1, S_2 of planar polygonal curves, a transformation t is searched for, that best maps S_1 to S_2. The classes of allowable transformations considered here are translations, homotheties (scaling and translation), rigid motions (rotation and translation), similarities preserving the direction of rotation (rotation, scaling and translation – without reflection), and general *affine maps*.

The intuition behind that matching is that features of the first set should be mapped into the proximity of corresponding features of the second set. If the images are not similar as a whole but do contain several independent subsets of corresponding features, there may not exist a single transformation but several transformations, each one matching only a subset of corresponding features. The course of the polygonal curves may be very helpful in identifying corresponding features. On the other hand different segmentations at crossings or different appearances of discontinuities may make this identification more difficult. Therefore a probabilistic voting scheme is applied here that uses votes of different weight and gathers them to form a set of candidate transformations.

In [8] we described a probabilistic matching approach related to the generalized Hough transform and briefly presented the idea of the new voting scheme which is described in detail here: During one random experiment a sample is a set of pairs of corresponding points from the two shapes. The quality of the match between two finite ordered point sets is measured by the weighted sum of quadratic distances between the corresponding points. The set of corresponding points is iteratively extended until no further data are available or the samples are no longer consistent. The resulting preliminary transformation is weighted with the quality of the match and the size of the matched point sets. 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 idea behind this, is that a transformation which gives a good match for the shapes, would give a good match for larger sets of points on these shapes. The details on selecting the point sets and computing the candidate transformations are presented in the following.

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.

2.1 Finding correspondences

Conspicuous features of curves arise from regions of high curvature [9] – regarding polylines these regions are the vertices. But not every vertex, even though its turning angle may be large, yields a feature recognizable by a human observer. For this reason, the mapping algorithm tries to find corresponding vertices but also may match a vertex without corresponding peer to a point lying on a line segment.

The initial part of every vote is the random choice of a single vertex in each of the two sets of polylines, and the direction for traversing the list of subsequent vertices (also both possible directions may be processed). The pairs of points are added to the set one by one. In each iteration step the matching transformation is updated. The final transformation for the set is weighted and handed over to the clustering algorithm.

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 subsequent vertices with respect to the randomly chosen direction. Analogous 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 subsequent vertices. The pair (p_0, q_0) is added to the – so far empty – sample set S.

In each iteration step the 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 other direction.

2.2 Calculating the transformations

For every new pair of vertices or vertex surrogates added to the sample set S a transformation is computed based on a least squares approach. The easiest way would be to compute the transformation that minimizes the sum of the squared distances of the vertices. This would favor parts with many vertices over parts with less vertices regardless of the extent and the expressiveness. In order to avoid this, we compute the transformation $t \in T$ that minimizes the sum of the weighted squared distances $\varepsilon(t) = \sum_{(p_i,q_i)\in S} w(p_i, q_i) \left\| q_i - t(p_i) \right\|^2$

with $w(p_i, q_i)$ being half the length of the edges incident to p_i and q_j.

If the class of the allowed transformations does not include scaling, which is the case for translations and rigid motions, we can determine vertex correspondences as described above and then compute the transformation minimizing the sum of weighted squared distances. However, if scaling is allowed, then the process of finding correspondences is no longer independent from the transformation, since the transformation could change distances between the vertices. We cope with this problem by using a prescaling factor s, which is randomly chosen such that $\mathrm{ld}(s)$ is normally distributed with mean value $\mathrm{ld}(\bar{s})$, where $\bar{s}$ is the prescaling factor of the transformation rated best so far with the initial value of $\bar{s}$ set to 1. S_1^s denotes the set S_1 scaled by s. 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^s.

Translations. The translation $t = (v_x, v_y)$ that minimizes the sum of the weighted squared distances $\varepsilon = \sum_{i=1}^{k} w(p_i, q_i) \left\| t(p_i) - q_i \right\|^2$ can easily be computed as $t = \frac{1}{w(S)} \sum_{i=1}^{k} w(p_i, q_i)(q_i - p_i)$ with $w(S)$ being the sum of all weights. That means that having computed the optimal transformation for the set $S' \subset S$ of cardinality $k-1$ the optimal transformation for the set S can be computed in constant time.

Homotheties. A homothety is defined by a scaling factor s and a translation vector $v = (v_x, v_y)$. A point p is mapped to $s \cdot p + v$. The optimal homothety is not computed directly, but first the prescaling is applied as described above, and then the optimal translation is computed in a second step. The scaling factor is restricted to be a positive real number, so that inversions are waived.

Rigid Motions. A rigid motion is defined by a rotation angle φ and a translation vector $v = (v_x, v_y)$. A point p is mapped to $Mp + v$ with M being the rotation matrix $\begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$.
For a set of pairs of (unweighted) points $S = \{(p_1, q_1), \ldots, (p_k, q_k)\}$ the rigid motion $t' = (\varphi', v')$ that minimizes the sum of the (unweighted) squared distances $\varepsilon' = \sum_{i=1}^{k} \left\| t(p_i) - q_i \right\|^2$ can easily be computed (as described in [10]):
Let $\bar{p} = \frac{1}{k} \sum_{i=1}^{k} p_i$, $\bar{q} = \frac{1}{k} \sum_{i=1}^{k} q_i$ denote the centers of mass of the sets of sample points, $\hat{p}_i = p_i - \bar{p}$, and $\hat{q}_i = q_i - \bar{q}$.
For $a = \sum_{i=1}^{k} (q_{ix} p_{ix} + q_{iy} p_{iy})$ and $b = \sum_{i=1}^{k} (q_{ix} p_{iy} - q_{iy} p_{ix})$ the rotation matrix is $M' = \frac{1}{\sqrt{a^2+b^2}} \cdot \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$.

The translation vector v' is determined as $v = \bar{q} - M \cdot \bar{p}$.

The rigid motion $t = (\varphi, v)$ that minimizes the sum of the weighted squared distances $\varepsilon = \sum_{i=1}^{k} w(p_i, q_i) \|t(p_i) - q_i\|^2$ can be computed using the method described above with slight modifications:
The centers of mass $\bar{p}$ and $\bar{q}$ are replaced by the weighted centers of mass. To compute the rotation according to the weighted squared distances instead of the squared distances, the radii (distance of a point to the weighted center of mass) are resized for that part. $\hat{p}_i = \sqrt{w(p_i, q_i)} \cdot (p_i - \bar{p})$, and $\hat{q}_i = \sqrt{w(p_i, q_i)} \cdot (q_i - \bar{q})$. With these new pairs of points, the conventional computation of M yields the desired result.
Detailed analysis shows, that all the terms can be transformed to avoid the explicit computation of the (weighted) centers of mass $\bar{p}$ and $\bar{q}$ such that having computed the optimal transformation for the set $S' \subset S$ of cardinality $k-1$ the optimal transformation for the set S can be computed in constant time.

Similarities. A similarity (preserving the direction of rotation) is defined by a rotation angle φ, a scale factor s, and a translation vector $v = (v_x, v_y)$. A point $p = (p_x, p_y)$ is mapped to $Mp + v$ with $M = \begin{pmatrix} s \cdot \cos\varphi & -s \cdot \sin\varphi \\ s \cdot \sin\varphi & s \cdot \cos\varphi \end{pmatrix}$.
First the prescaling is applied as described above, and then the optimal similarity is computed in a second step.

For a set of pairs of points $S = \{(p_1, q_1), \ldots, (p_k, q_k)\}$ the similarity $t = (M, v)$ that minimizes the sum of the weighted squared distances $\varepsilon = \sum_{i=1}^{k} w(p_i, q_i) \cdot \|t(p_i) - q_i\|^2$ can easily be computed by solving the following system of linear equations:

$$\sum_{i=1}^{k} w(p_i, q_i) \begin{pmatrix} {p_{ix}}^2 + {p_{iy}}^2 & 0 & p_{ix} & p_{iy} \\ 0 & {p_{ix}}^2 + {p_{iy}}^2 & p_{iy} & -p_{ix} \\ p_{ix} & p_{iy} & 1 & 0 \\ p_{iy} & -p_{ix} & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} m_{11} \\ m_{12} \\ v_x \\ v_y \end{pmatrix}$$

$$= \sum_{i=1}^{k} w(p_i, q_i) \begin{pmatrix} q_{ix}p_{ix} + q_{iy}p_{iy} \\ q_{ix}p_{iy} - q_{iy}p_{ix} \\ q_{ix} \\ q_{iy} \end{pmatrix}$$

That means that having computed the optimal transformation for the set $S' \subset S$ of cardinality $k-1$ the optimal transformation for the set S can be

computed in constant time.

Affine Maps. An affine transformation is defined by an arbitrary matrix $M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$ and a translation vector $v = (v_x, v_y)$. A point $p = (p_x, p_y)$ is mapped to $Mp + v$. First the prescaling is applied as described above, and then the optimal affine map is computed in a second step.

For a set of pairs of points $S = \{(p_1, q_1), \ldots, (p_k, q_k)\}$ the affine map $t = (M, v)$ that minimizes the sum of the weighted squared distances $\varepsilon = \sum_{i=1}^{k} w(p_i, q_i) \cdot \|t(p_i) - q_i\|^2$ can easily be computed by solving the following system of linear equations:

$$\sum_{i=1}^{k} w(p_i, q_i) \begin{pmatrix} p_{ix}p_{ix} & p_{ix}p_{iy} & p_{ix} & & & \\ p_{ix}p_{iy} & p_{iy}p_{iy} & p_{iy} & & 0 & \\ p_{ix} & p_{iy} & 1 & & & \\ & & & p_{ix}p_{ix} & p_{ix}p_{iy} & p_{ix} \\ & 0 & & p_{ix}p_{iy} & p_{iy}p_{iy} & p_{iy} \\ & & & p_{ix} & p_{iy} & 1 \end{pmatrix} \cdot \begin{pmatrix} m_{11} \\ m_{12} \\ v_x \\ m_{21} \\ m_{22} \\ v_y \end{pmatrix}$$

$$= \sum_{i=1}^{k} w(p_i, q_i) \begin{pmatrix} q_{ix}p_{ix} \\ q_{ix}p_{iy} \\ q_{ix} \\ q_{iy}p_{ix} \\ q_{iy}p_{iy} \\ q_{iy} \end{pmatrix}$$

If the points are collinear, the affine map is not uniquely defined and the system of linear equations has no unique solution. In this special case, the optimal similarity is taken instead.
Again, having computed the optimal transformation for the set $S' \subset S$ of cardinality $k - 1$ the optimal transformation for the set S can be computed in constant time.

In each iteration step it is checked whether the error introduced by the new added pair is still within tolerance bounds. We define the maximum tolerated error for a sample as a linear function of the perimeter of the bounding box containing the sample. The perimeter of the bounding box is weighted by a constant parameter called relative error threshold. If the error introduced by the last added pair is too big, the traversal of the polylines is ceased, as illustrated in Figures 1 and 2: 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).

This definition of the stop criterion and the choice of the best index are invariant under scalings and can be done in constant time. To achieve invariance under rotations also, the bounding box had to be replaced by the minimum enclosing circle.

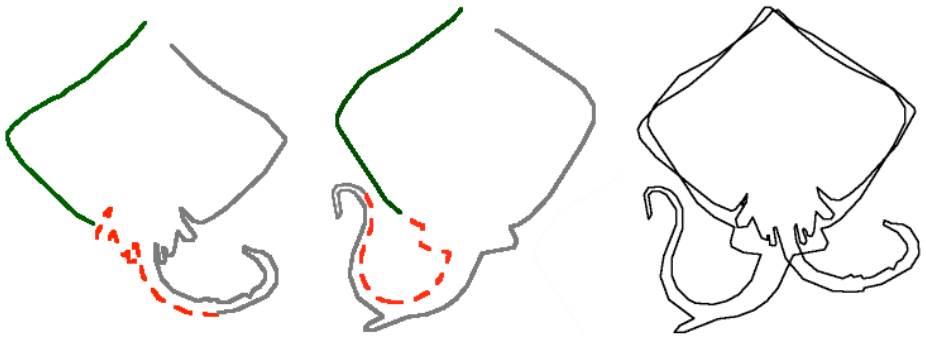


Fig. 1. Two instances of the MPEG-7 shape B data set (ray-7, ray-20 and both mapped).

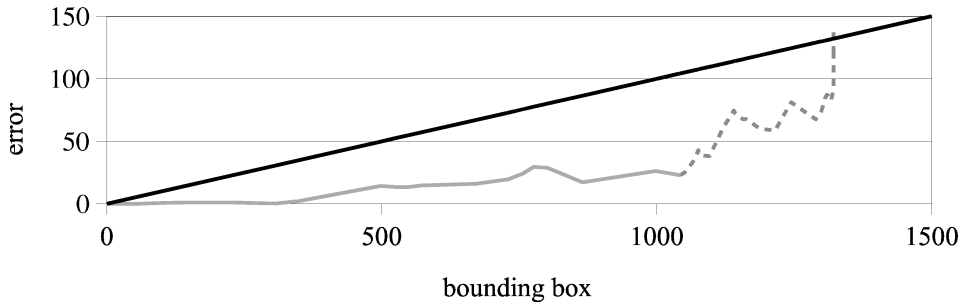


Fig. 2. Perimeter of the bounding box vs. error of last added pair of points. The part of the polyline defining the transformation for this vote is plotted as a solid light grey line, the part skipped is plotted dashed. The solid dark line is the maximum tolerated error bound.

2.4 Weighting the transformations

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.

The most common technique for the clustering of transformations (often referred to as pose clustering) – histogramming the transformations in the multidimensional transformation space (see [6]) – discards the effects on the transformed shapes. Two rotations may yield nearly the same results if applied to a shape with its center being the origin, or totally different results if the shape's center is far away from the origin. To avoid this imbalance, a distance measure for transformations is used here, which considers the shapes' properties. Let t_1 and t_2 be two arbitrary transformations and let S_1 be the transformed shape. The distance measure $d_{S_1}(t_1, t_2) = \max_{p \in S'} \|t_1(p) - t_2(p)\|$, with S' being the set of the vertices of the bounding box of S_1 forms a metric space for affine maps, under the assumption that the four points of S' are pairwise different. The distance between two transformations depends, thus, on the shape to be transformed and reflects the difference in the image of the shape under the considered transformations.

2.5 Clustering

A cluster in our sense is a region of limited diameter, which subsumes a considerable amount of weight of the enclosed input points (transformations). In the clustering process we want to find all clusters with large weight because they give evidence of good matching transformations.

Let T_n be the set of n transformations generated by n random experiments and W_i be the weight of a transformation $t_i \in T_n$. For a fixed cluster radius r_c a cluster C_t with center $t \in T_n$ is defined as the set $\{t_i \in T_n | d(t_i, t) < r_c\}$ that is the set of transformations with distance less than r_c to the center. The weight of a cluster is defined as the sum of the weights of its elements. This definition is related to what is called *naive density estimator* in statistics.
The transformations that are considered as center of a cluster are identified

as follows: $t_i \in T_n$ is called *dominator* of $t_j \in T_n$ if and only if $d(t_i, t_j) < r_c$, $W_i > W_j$, and no other transformation is dominator of t_i. Each transformation $t \in T_n$ that has no dominator is the center of a cluster C_t. In other words: a transformation t either is the center of a cluster or it is contained in at least one cluster of its dominators. This definition allows for a fast computation of all clusters and their weights.

The clusters may be determined by iteratively taking the transformation with highest weight as center of a cluster, removing the cluster's members from the set of potential centers and continuing with the reduced set. A naive algorithm would need time quadratic in the number of transformation. This can be decreased by partitioning the transformation space and organizing it in a tree structure.

Every node of the tree may store a cluster which stores the transformations belonging to it. A node u_i^j on level i with index j represents a ball with some radius r_i around its center (its cluster's center). It may have arbitrarily many children, each one representing a ball with radius $r_{i+1} = r_i/2$ and a center that lies inside the ball represented by u_i^j, see Figure 3 for a schematic illustration. The root node represents a ball with radius r_0 containing all sample transformations. The smallest radius in the tree is the given cluster radius r_c. The children of a node are ordered, each one only responsible for the part of the space not covered by its preceding siblings.

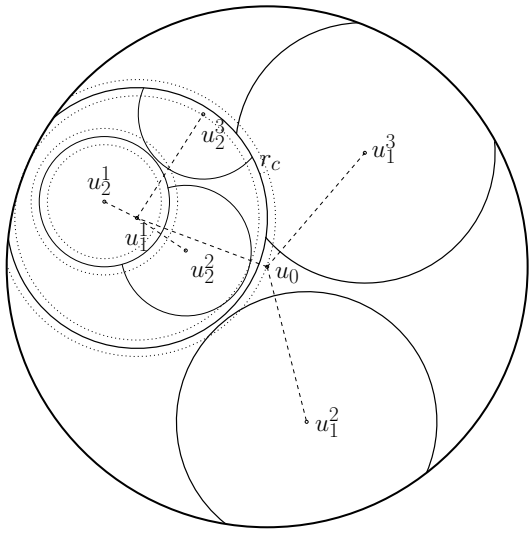


Fig. 3. Partition of the transformation space.

The center c_0 and the radius r_0 of the ball represented by the root node may be easily computed for the classes of transformations that do not allow scalings: Let c_{bb1}, c_{bb2} denote the centers and D_{bb1}, D_{bb2} denote the diameter of the bounding boxes of the shapes S_1 and S_2 respectively. Then c_0 is chosen to be the translation defined by $c_{bb2} - c_{bb1}$. Any transformation t generated by the

random experiments will fulfill the condition that the transformed bounding box of S_1 at least touches the bounding box of S_2. Therefore $d_{S_1}(t, c_0) \leq D_{bb1}/2 + D_{bb2}/2 + D_{bb1}$.

For the classes of transformations that do allow scalings the space is not bounded in such a natural way. However, if the application provides a bound on the maximum scaling factor s_{max} the distance between any transformation t (homothety or similarity) and c_0 can be bounded in a similar way: $d_{S_1}(t, c_0) \leq D_{bb1}/2 + D_{bb2}/2 + s_{max}D_{bb1}$. If no such bound is given, the root node and its radius can be updated during the construction of the tree.

The clustering is performed as follows: The sample transformations are sorted according to their weight and they are processed in descending order. Beginning from the empty tree with root node (c_0, r_0) in each iteration step a transformation t is added to the tree. First, the tree is searched for all clusters such that t is contained in a ball of radius r_c around the center of the cluster. If such clusters exist, then t is added to all these clusters. If no cluster containing t is found, a new cluster C with center t is created and inserted into the tree.

When a node with center t_u and radius r is searched for clusters neighboring a transformation t, the distance $d(t_u, t)$ is computed. If $d(t_u, t) < r - r_c$, all the clusters worth considering have to lie inside the node's ball and the subsequent siblings of the node may be discarded. If $d(t_u, t) > r + r_c$ the clusters have to lie outside and the node u may be discarded. In the other cases the node and the subsequent siblings have to be considered. The search is then performed recursively in all nodes, that may contain t.

For a cluster C being inserted into a node representing a ball with radius r, if there exists at least one child node representing a ball containing the center of C, C is recursively inserted into the first such child. Otherwise, a new child holding C with radius $r/2$ and a center corresponding to the center of C is created and appended to the list of child nodes.

After having processed all transformations, the clusters are sorted according to their weights. The clusters with the highest weights provide the candidate transformations. The number of candidate transformations may be chosen as a constant or we may consider the clusters with weight up to a certain fraction of the maximum weight.

Properties of the tree. Since the number n of samples is finite, the considered transformation space T_i is bounded, i.e. $\exists r_i \in \mathbb{R} : \forall x, y \in T_i : d(x, y) < r_i$. A subset $C_i \subset T_i$ is called an ε-packing if and only if $\forall x, y \in C_i : d(x, y) > 2\varepsilon$. The size of the largest ε-packing is called the packing number $P(T_i, \varepsilon)$. For $\varepsilon \to 0$, $P(T_i, \varepsilon) = O\left(\left(\frac{r_i}{\varepsilon}\right)^{\mathcal{D}}\right)$ for $\mathcal{D}$ being the dimension of the space [11].

Therefore the maximum number of balls of radius $r_i/2$ with centers in T_i such that no center is contained in another ball, is in $O(2^{\mathcal{D}})$. This means that the number of children a node of the tree may have is bounded by a constant only depending on the dimension of the transformation space. The depth of tree is at most $\lceil \lg(r_0/r_c) \rceil$.

3 The similarity function

After some candidate transformations have been found, a distance or similarity measure has to be applied to rate the similarity of the two matched shapes. Most of the existing distance measures are either not applicable to the sets of polygonal curves (like Fréchet distance or turning angle function) or being a maximum based distance (like Hausdorff distance) are too sensitive to noise. We describe a new similarity measure which averages over the whole set of polylines, so it is not sensitive to noise, but looses the property of being a metric. It takes into account special properties of line segments, and is invariant to different parameterizations or splitting of polylines.

The resemblance function is defined for every point of the polylines and stands for how good the point is represented by the other set. It is composed of the point's distance to the points of the other shape and of the similarity of slopes.

Let h be a straight line segment of the first set S_1 with endpoints p_0 and $p_0 + v$, and g a segment of the second set S_2 with endpoints q_1, q_2. Let h' and g' denote the supporting lines of the segments h and g respectively. For a point $p \in h$ we define the distance to g as the distance to a point q on g, such that the orthogonal projection of q on h is exactly p, if such a point q exists. Otherwise, if p' is the nearest orthogonal projection of an endpoint of g on h', the the distance from p to g is defined as the distance from p to p' plus the distance from p' to the corresponding endpoint of g. Formally, consider a parameterization of h': $p(\lambda) = p_0 + \lambda \cdot v$, $\lambda \in \mathbb{R}$. Let $q(\lambda)$ denote a point on g', such that $p(\lambda)$ is an orthogonal projection to h' of $q(\lambda)$. Further, let $p_1 = p_0 + \lambda_1 \cdot v$ and $p_2 = p_0 + \lambda_2 \cdot v$ denote projections of the endpoints q_1 and q_2 on h', and w.l.o.g., let $\lambda_1 < \lambda_2$, see Figure 4 for an illustration. The distance function is then defined as

$$\delta_{h,g}(\lambda) = \begin{cases} \|p_1 - q_1\| + \|p(\lambda) - p_1\|, & 0 \le \lambda < \lambda_1 \\ \|p(\lambda) - q(\lambda)\|, & \lambda_1 \le \lambda \le \lambda_2 \\ \|p_2 - q_2\| + \|p(\lambda) - p_2\|, & \lambda_2 < \lambda \le 1 \end{cases}$$

This definition of the distance (unlike the Euclidean distance) ensures that the function $\delta_{h,g}(\lambda)$ is piecewise linear, which allows for a fast computation of

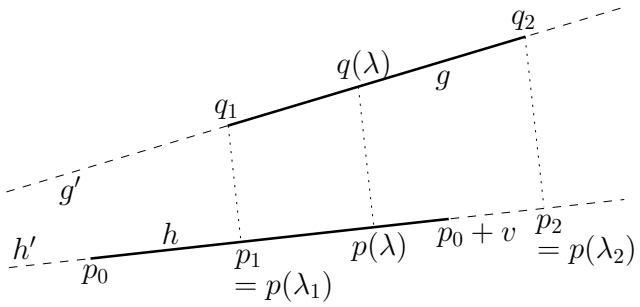


Fig. 4. Definition of the distance between two line segments

the resemblance function.

If the distance of a point $p \in h$ and the segment g equals zero, that is, p lies on g, then we say that p is exactly represented by g – the degree of being represented, therefore, is 1. The greater the distance gets, the lesser p is represented by g. The decrease in similarity is weighted with the size of the shape S_1. In [12] an *inverse distance function* is used in a similar context for the rating of transformations in an optimization problem. For their task they chose a function that exponentially decreases with higher Euclidean distance to value the correspondence of features (see Figure 5(a)).

In the present case the goal is not to find an optimum, but to rate a given configuration. Small deviations in the position of the features of the two sets should not result in an excessive decrease in similarity function. Therefore an inversion function with a high (negative) slope around the y-axis is inapplicable. The function $\alpha'_{h,g}(\lambda) = \exp(25(\delta_{h,g}(\lambda)/D_{S_1})^2)$, where D_{S_1} denotes the diameter of the shape S_1, seems more promising. It rates pairs with a distance less than 5 % of the diameter very high (over 0.9) and with a distance of more than 25 % of the diameter very low – around 0.2 (see Figure 5(b)). To make the computation easier, the piecewise quadratic function

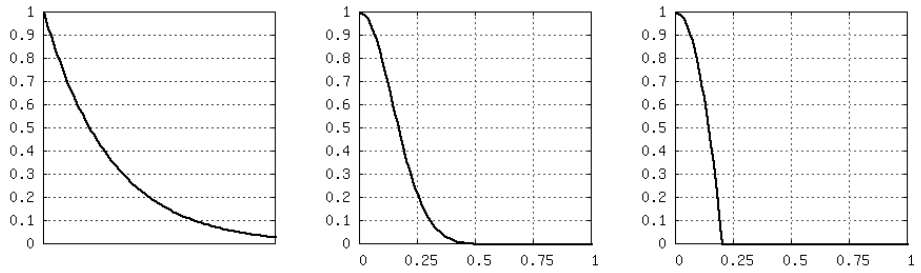
$$\alpha_{h,g}(\lambda) = \max(1 - 25\left(\frac{\delta_{h,g}(\lambda)}{D_{S_1}}\right)^2, 0)$$

is chosen. It has the same characteristics for small distances (up to 10 %) but decreases faster for greater distances (see Figure 5(c)).

The resemblance of two line segments also depends on their *slopes*. Line segments with similar slopes should get a higher resemblance value, so a slope factor $\beta_{h,g}$ is defined as

$$\beta_{h,g} = \cos\left(\angle(h,g)\right)^4$$

It rates pairs with a difference in slopes of less than 10° very high (over 0.9) and with a difference of more than 45° very low (below 0.25). The exponent 4 was chosen experimentally.



(a) (b) (c)

Fig. 5. (a) exponentially decreasing inverse distance; (b) inversion function α'; (c) inverse distance function α

The *resemblance function* ϕ_h for a line segment h is defined as a combination of inverse distance function and slope rate:

$$\phi_h(\lambda) = \max_{g \in S_2} (\alpha_{h,g}(\lambda) \cdot \beta_{h,g}) \tag{1}$$

In applications that follow human perception, parts with many parallel line segments have to be prevented from dominating over parts with solitary line segments. Therefore a weight function ω is defined analogously to the resemblance function. It rates the density of similar line segments of an image. For a line segment $h \in S_1$ it is defined as

$$\omega_h(\lambda) = \frac{1}{\sum_{g \in S_1} (\alpha_{h,g}(\lambda) \cdot \beta_{h,g})} \tag{2}$$

Note, that the weight function rates the similarity of a segment h to the other segments in the same set.

The directed resemblance measure for two sets of line segments $\Phi_{\rightarrow}(S_1, S_2)$ is defined as a weighted mean over all points of the shape S_1:

$$\Phi_{\rightarrow}(S_1, S_2) = \frac{\sum_{h \in S_1} \left(\int_{\lambda=0}^{1} \phi_h(\lambda) \cdot \omega_h(\lambda)\, d\lambda \cdot l_h \right)}{\Omega(S_1)}, \tag{3}$$

with l_h being the length of h and $\Omega(S_1)$ being the total weight of S_1: $\Omega(S_1) = \sum_{h \in S_1} \left(\int_{\lambda=0}^{1} \omega_h(\lambda)\, d\lambda \cdot l_h \right)$.

The undirected resemblance measure $\Phi(S_1, S_2)$ is defined as the weighted arithmetic mean:

$$\Phi(S_1, S_2) = \frac{\Phi_{\rightarrow}(S_1, S_2) \cdot \Omega(S_1) + \Phi_{\rightarrow}(S_2, S_1) \cdot \Omega(S_2)}{\Omega(S_1) + \Omega(S_2)}. \tag{4}$$

From this resemblance measure a deviation or distance measure may be derived, but of course this will never be a metric as the triangle inequality does

not hold.

Computational complexity. The resemblance value is computed evaluating the integrals of a combination of the resemblance function and the weighting function for every line segment. For two sets with n line segments each, the resemblance function – as defined in Equation (1) – for a single line segment is the upper envelope of at most $4 \cdot n + 1$ regular (partially defined) functions. Using quadratic functions, each pair intersects at most 2 times (unless equal). According to the upper bound on the length of Davenport-Schinzel sequences [13] the complexity of the upper envelope of the $4 \cdot n + 1$ functions is bounded by $O(n \cdot 2^{\alpha(n)})$ with α being the inverse Ackermann function.

The weighting function for a single line segment – as defined in Equation (2) – is the sum of n functions, each one split into at most 4 regular pieces. The number of intervals for the sum is at most $3n + 1$. So the overall complexity for all the line segments is bounded by $O(n^2 \cdot 2^{\alpha(n)})$.

3.1 Partial similarity function

For the *complete-complete matching* resemblance function as defined in Equation (4) we took a weighted combination of two one-sided resemblance values. Note, that the definition of the directed resemblance function (Equation (3)) applied to the complete shapes S_1 and S_2 gives us a valuation of how good the complete shape S_1 is matched to shape S_2 and, therefore, a valuation for the *complete-partial matching*.

For *partial-partial matching* we keep for each cluster a record of which parts of the shapes contribute to the transformations contained in this cluster. Let C be a cluster and $S_1^C \subset S_1$ and $S_2^C \subset S_2$ are the parts of the shapes that contributed to the transformations in C. Then, we compute the resemblances for the matched parts: $s_1 = \Phi_{\rightarrow}(S_1^C, S_2^C)$ and $s_2 = \Phi_{\rightarrow}(S_2^C, S_1^C)$. These values waive the remaining parts S_1
S_1^C and S_2
S_2^C of the shapes completely, so in general the highest values would be achieved by clusters matching very small parts perfectly.

The size of the matched parts also has to affect the value of partial similarity. Therefore, we compute a ratio of the matched parts as $\rho_1 = \frac{\left|S_1^C\right|}{|S_1|}$ and $\rho_2 = \frac{\left|S_2^C\right|}{|S_2|}$ respectively where $|S_i|$ denotes the total length of the polylines of the shape S_i and $\left|S_i^C\right|$ denotes the length of the parts matched contributing to the cluster C. A weight factor $f_i = 1 - (1 - \rho_i)^k$ (see Figure 6) if defined, where k is a (user-defined) parameter; the default value of k in our implementation is 3.

The maximum of two weight factors $f^* = \max(f_1, f_2)$ is then used to adjust the resemblance value: $s^* = f^* \frac{s_1+s_2}{2}$.

The choice of the factor function was motivated by the following consideration: if large parts of at least one shape are matched, we want to leave the resemblance value almost unchanged, and give larger penalties the smaller the matched parts get. With the parameter k the user can control these penalties, if k is large, the resemblance value stays almost unchanged even for small parts, whereas for small values of k the quality of match decreases with the relative size of matched parts.

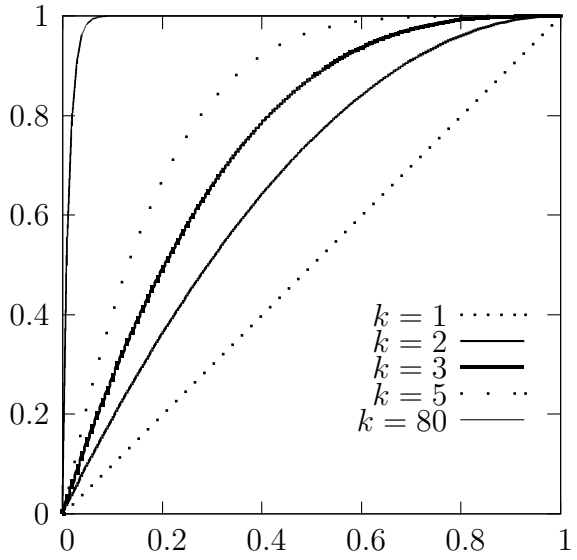


Fig. 6. Weight factor function parameterized by k.

4 Experimental results

We implemented the matching algorithms and the resemblance measure as an automated application that finds the best resemblance value for every pair of shapes from a given set of shapes. The "CE-Shape-1" part B dataset from the MPEG-7 shape silhouette database was used as test data. It consists of 1400 (mostly) silhouette images, subdivided into 70 classes containing 20 related images each. From the images the outer closed contours were extracted. The polylines for which every vertex corresponds to a pixel, were then simplified using the Douglas-Peucker algorithm [14].

The resemblance of the shapes was tested under similarity transformations including reflections. To avoid unnecessarily many unsuccessful attempts, the shapes were scaled in advance so that their bounding boxes had the same

diameter. The whole comparing process was done repeatedly, 3 times with the original (pre-scaled) shape and 3 times with one shape flipped to incorporate reflections. As result of the comparison of two shapes the highest resemblance value encountered for any of the candidate transformations and for any of the iterations was taken.

Every shape was compared to all the 1400 shapes of the set, including itself, and the nearest neighbors, that is shapes with highest resemblance value, were determined. The performance was rated based on three values:
True Positives as Nearest Neighbors: the ratio of shapes from the same class found as consecutive first nearest neighbors. The average for all the 1400 shapes was 67.78%.
True Positives in Class Size: the ratio of shapes from the same class found among the 20 nearest neighbors. The average for all the 1400 shapes was 76.89%.
True Positives in Double the Class Size: the ratio of shapes from the same class found among the 40 nearest neighbors, the so called bull's eye performance. The average for all the 1400 shapes was 84.28%. The best bull's eye performance of 84.33% on the MPEG-7 shape silhouette database was reported by Attalla and Siy in [15].

We also evaluated our system using a collection of 10 745 abstract images from the UK Trade Marks Registry and a set of 24 image queries. This is the same test set as was used for the evaluation of the Artisan system as reported in [16]. The set of relevant images for each query was selected by experienced trademark examiners and was used as a benchmark for the system evaluation.

We evaluated the performance of our system on the trademark image set according to the performance measures used in [16]: *normalized recall* R_n, *normalized precision* P_n and *normalized last place* L_n, which are defines as

$$
\begin{aligned}
R_n &= 1 - \frac{\sum_{i=1}^{n} R_i - \sum_{i=1}^{n} i}{n(N-n)} \\
P_n &= 1 - \frac{\sum_{i=1}^{n} (\log R_i) - \sum_{i=1}^{n} (\log i)}{\log\left(\frac{N!}{(N-n)!n!}\right)} \\
L_n &= 1 - \frac{R_l - n}{N-n},
\end{aligned}
$$

where n is the total number of the relevant images, N is the size of the whole collection, R_i is the rank at which relevant image i is actually retrieved, and R_l is the rank at which the last relevant image is retrieved. All three measures rank a system's retrieval performance in response to a query from 0 to 1, with 1 meaning perfect retrieval. The major difference between normalized recall and precision is that normalized recall gives higher weighting to success in retrieving the first few items, while normalized precision gives equal weighting

to all retrievals. The last place ranking indicates the number of retrieved items a user has to search in order to have a reasonable expectation of finding all relevant items. This measure is useful for applications requiring an exhaustive search, for example trademark retrieval.

The performance achieved by our system on the trademark test set is: normalized recall of 0.93, normalized precision of 0.71, normalized last place of 0.68. The early implementation of the Artisan system (which is regarded as one of the most comprehensive trademark retrieval systems in the current literature [17]) had the values of 0.90, 0.63, and 0.56, respectively.

The experiments show that the algorithm is robust with respect to noise and to differences in the representation of the shapes. However, apart from the good results achieved, we recognized some cases – especially among the trademark images – that are problematic for our approach:

- frames
  If the important part of a trademark image is surrounded by some kind of a simple frame, most humans do not pay much attention to that frame. The similarity measure however is influenced by it, because the frames naturally are larger than the part contained in it. To tackle this problem by using the partial-partial matching variant may result in high similarity values for completely different logos just because the frames are identical.
- spatially independent parts
  Comparing two images, that consist of two or more spatially independent parts and the corresponding parts are similar but arranged in slightly different ways, most humans do not care about the differences. However, there exists no affine map that aligns all parts properly at the same time.

To overcome these problems will be part of our future work.

We think the results achieved on both test sets are highly encouraging. They indicate that our method is a general purpose matching technique, not limited to contour shapes, but also performs well on complex shapes within the context of the non-trivial task of trademark image retrieval.

5 Conclusions

In this paper we presented a shape matching algorithm that randomly selects a point sample in each shape and gives a vote to a transformation which maps one random sample to the other minimizing the squared distances between the corresponding points. Instead of selecting a minimum size sample for the given class of transformations, as it is usual in voting based methods, we ex-

tend the samples until the whole data is incorporated or the samples are no longer consistent. The transformation matching the sample sequences is then weighted according to the quality of match and the size of the samples. After sufficient number of random experiments the weighted votes in transformation space are clustered and the clusters with high total weight are taken as candidate transformations.

The second part of our method is similarity evaluation. Each candidate transformation is applied to one shape and the resemblance of the two shapes is rated according to the distance between the points of two shapes and to the similarity in slopes of the straight line segments contained in the shapes. We also define complete-partial similarity variant of our resemblance function, which reflects how similar the complete shape S_1 is to some parts of the shape S_2, and a partial-partial similarity variant, i.e., how good a part of shape S_1 matches some part of shape S_2.

We applied the implementation of our algorithms to the "CE-Shape-1" part B dataset from the MPEG-7 shape silhouette database, and to a test collection of 10 745 trade mark images provided by the UK Trade Marks Registry with a set of 24 image queries, both with convincing results.

The challenges mentioned in section 4 may be tackled by dividing the images into meaningful parts, weighting them, and applying the matching and similarity evaluation as presented in this paper to this parts.

References

[1] H. J. Wolfson, I. Rigoutsos, Geometric hashing: An overview, IEEE Computational Science and Engineering 04 (4) (1997) 10–21.

[2] D. Huttenlocher, S. Ullman, Recognizing solid objects by alignment with an image, International Journal of Computer Vision 5 (2) (1990) 195–212.

[3] R. C. B. Martin A. Fischler, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (1981) 381–395.

[4] A. S. Aguado, E. Montiel, M. S. Nixon, Invariant characterisation of the hough transform for pose estimation of arbitrary shapes, Pattern Recognition 35 (2002) 1083–1097.

[5] S. Moss, E. R. Hancock, Pose clustering with density estimation and structural constraints, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, 1999, pp. 2085–2091.

[6] C. F. Olson, Efficient pose clustering using a randomized algorithm, Int. J. Comput. Vision 23 (2) (1997) 131–147.

[7] G. Stockman, Object recognition and localization via pose clustering, Computer Vision, Graphics, and Image Processing 40 (1987) 361–387.

[8] H. Alt, L. Scharf, S. Scholz, Probabilistic matching of sets of polygonal curves, in: Proceedings of the 22nd European Workshop on Computational Geometry (EWCG), Delphi, Greece, 2006, pp. 107–110.

[9] F. Attneave, Some informational aspects of visual perception, Psychological Review 61 (3) (1954) 183–193.

[10] J. H. Challis, Estimation of the finite center of rotation in planar movements, Medical Engineering & Physics 23 (3) (2001) 227–233.

[11] K. L. Clarkson, Nearest-neighbor searching and metric space dimensions, in: G. Shakhnarovich, T. Darrell, P. Indyk (Eds.), Nearest-Neighbor Methods for Learning and Vision: Theory and Practice, MIT Press, 2006, pp. 15–59.

[12] A. Pinz, M. Prantl, H. Ganster, A robust affine matching algorithm using an exponentially decreasing distance function, Journal of Universal Computer Science 1 (8) (1995) 614–631.

[13] P. K. Agarwal, M. Sharir, Davenport-Schinzel sequences and their geometric applications, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000, pp. 1–47.

[14] D. Douglas, T. Peuker, Algorithms for the reduction of the number of points required to represent a digitised line or its caricature, in: The Canadian Cartographer, Vol. 10, 1973, pp. 112–122.

[15] E. Attalla, P. Siy, Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching, Pattern Recognition 38 (12) (2005) 2229–2241.

[16] J. P. Eakins, J. M. Boardman, M. E. Graham, Similarity retrieval of trademark images, IEEE MultiMedia 5 (2) (1998) 53–63.

[17] H. Jiang, C.-W. Ngo, H.-K. Tan, Gestalt-based feature similarity measure in trademark database., Pattern Recognition 39 (5) (2006) 988–1001.