

PROFI



Project number:	FP6-511572
Project acronym:	PROFI
Title:	Perceptually-relevant Retrieval Of Figurative Images

Deliverable No: D5.3:	Algorithms for partial matching of curves
-----------------------	---

Short description:

Randomized algorithms are developed for finding similarities between parts of two shapes A and B . Shapes are modeled by sets of line segments or polygonal curves. The major idea is the same as for complete matching: to take random samples of points from both shapes and give a “vote” for that transformation (translation, rigid motion, or similarity) matching one sample with the other. If that experiment is repeated frequently we obtain by the votes, a certain distribution of points in the space of transformations. Clusters of this point set indicate which transformations give the best match between the two figures.

Due month:	12
Delivery month:	12
Lead partner:	FUB
Partners contributed:	FUB, UU
Classification:	Public



Project funded by the European Community under the “Information Society Technologies” Programme

1 Introduction

In this work package we develop algorithms for *matching* parts of two planar shapes. We assume that shapes are modeled by sets of plane polygonal curves (or sets of line segments). As possible classes of transformations we will consider translations, rigid motions (i.e., translations and rotations) and similarities (i.e., translations, rotations and scalings).

The general situation is that we are given two objects A and B and a set T of allowable transformations and we want to transform B optimally so that the transformed image of B , or some part of it, is as close to A , or a part of A , as possible. Usually the quality of match is measured by some distance function or similarity measure $\delta(A, B)$ which assigns a number to any pair of objects A and B . We call the problem addressed in work package four of optimal matching of the complete shape B to the complete shape A a *complete-complete matching* (CCM). In this work package we consider the problem of *complete-partial matching* (CPM), i.e., matching B completely as good as possible to some part of A , and *partial-partial matching* (PPM), i.e. matching some part of B as good as possible to some part of A . Clearly, CPM and PPM are not uniquely specified since there is a tradeoff between the quality of the match and the size of the parts that match. Which of the two criteria is more important, depends on the application. In case of trademarks we expect partial-partial matching, where the parts are “not too small” and the quality of match is “fairly good”, to depict best the perceptual similarity.

Several similarity measures and algorithms are known to match two curves, especially polygonal curves, see [2, 3] for surveys or the deliverable report 4.1 for a short overview. Some of these similarity measures can be modified to valuate partial match, e.g. percentile-based Hausdorff distance [2].

The probabilistic method described in deliverable report 4.1 is applicable for the partial matching problem as well. Furthermore, if there are several parts of the shape B that match some parts of A under different transformations all those transformation would be found among the candidate transformations determined by the probabilistic algorithm.

2 Results

2.1 Probabilistic partial matching of sets of line segments

Given two sets $A, B \subset \mathbb{R}^2$ consisting of all points lying on a finite set of polygonal curves (or, equivalently, line segments). Find that transformation t , which lets parts of the transformed image of B , $t(B)$, *match best* A , i.e. puts the most similar parts of the shapes A and B on top of each other. As sets of allowed transformations we consider translations, rigid motions, that is translations and rotations, and similarities, i.e. translations, scalings and rotations. These transformation spaces are two-, three- and four-dimensional respectively.

The algorithm described in report 4.1 is quite simple:

1. Take an appropriate random sample S_A of A and a random sample S_B of 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 sampling points in their δ -neighborhood as candidates for good transformations.

The size of the random sample depends on the allowed transformations. For *translations* it contains one point: $S_A = \{a\} \subset A$, $S_B = \{b\} \subset B$, and the corresponding translation is $t = a - b$. The points are selected randomly under uniform distribution as described in report 4.1.

A *rigid motion* is uniquely defined by a point-vector pair, therefore each random sample contains a point of a shape and a unit length direction vector: $S_A = \{(a, d_a) : a \in A, d_a \in \mathbb{R}^2\}$, $S_B = \{(b, d_b) : b \in B, d_b \in \mathbb{R}^2\}$. The sampling points are selected randomly and the direction vector is a kind of approximation of a tangent at the sampling point. Several methods of selecting a direction vector are discussed in report 4.1. The corresponding rigid motion is defined by a rotation α , that maps the vector d_b to d_a , and a translation $a - \alpha(b)$ that maps a rotated point b to the point a .

For *similarity maps* random samples contain each two points of the corresponding shapes: $S_A = \{a_1, a_2\} \subset A$ and $S_B = \{b_1, b_2\} \subset B$, which define uniquely a similarity map s such that $s(b_1) = a_1$ and $s(b_2) = a_2$.

The analysis of the probability distribution and the error bounds on the approximation of the distribution in the case of translations can be found in report 4.1. Here we describe the application of this probabilistic method to the partial matching problem. Let $p(t)$ denote the probability distribution of a δ -neighborhood of transformation t . This probability value is estimated in our algorithm by the ratio of sampling points in the δ -neighborhood of the transformation t to the total number of sampling points. As discussed in the report 4.1 the value $p(t)$ is proportional to the size of the parts of the shapes A and B brought by the transformation t into the δ -neighborhood of each other.

The choice of δ controls the quality of the match. If we want to find nearly congruent parts in two figures, we should use a small value of δ , and if a rough match is wanted a larger value of δ gives the desired results. Furthermore, we should examine several local maxima of the $p(t)$ function, since each of the maxima corresponds to a partial match between two figures. For each of the candidate transformations we then can determine which parts of the two shapes match and how large these parts are.

This step can be performed using the directed Hausdorff distance. The directed Hausdorff distance from a compact set B to a compact set A assigns to every point in B the distance to its nearest neighbor in A and takes maximum over all such distances. For the sets of line segments in the plane an $O(n \log n)$ algorithm for computing the Hausdorff distance is given in [1], where n is the number of segments in the sets.

Then 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 . Using the algorithm from [1] we can perform this verification step in time $O((n+m) \log(n+m))$, where n and m are the number of segments in A and B , respectively. Alternatively we can use the similarity measure proposed in the next section in a similar way. In this case the verification step takes $O(nm)$ time.

As we mentioned above 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. After the verification step we can report the actual similarity in case it is better than the predefined.

Figure 1 shows an example where there are two good positions for the shape B corresponding to a complete-partial match. Both of those positions correspond to a local maximum of the distribution in the translation plane. Of course, there are more than two local maxima,

so we should consider them up to a certain threshold, e.g. up to 50% of the largest value.

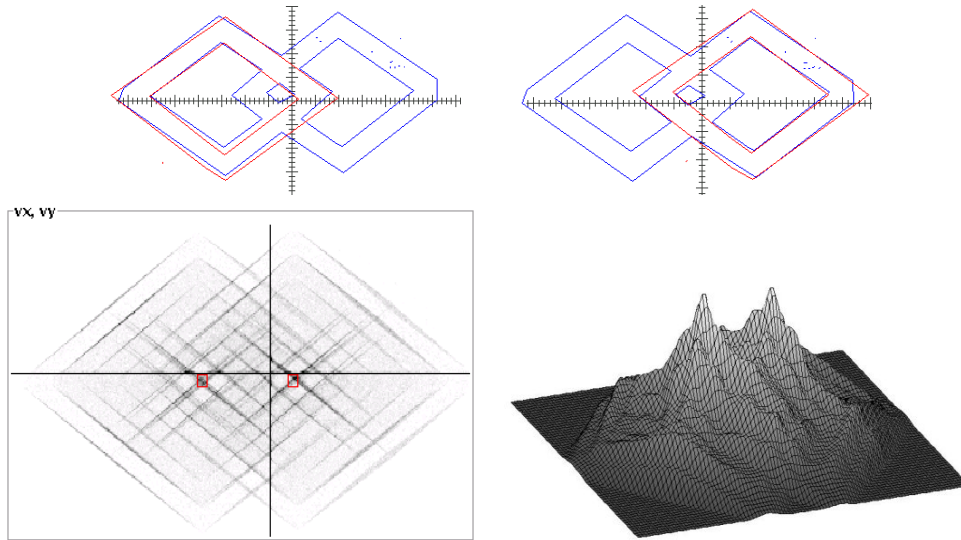


Figure 1: Top: two superimpositions of the figures each corresponding to a complete-partial match. Bottom: Experimental distribution in the translation space with two local maxima marked and a 3d-view of the distribution.

Another example of a complete-partial matching is shown in Figure 2. Here the superimposition of the horse shape in figure *B* with the part of the shape *A* depicting a horse corresponds to a single significant local maximum of the probability distribution in the translation plane.

2.2 Probabilistic partial matching of sets of polygonal curves

The heuristic computing optimal transformations for finite ordered sets of polyline vertices as described in report 4.1 is not limited to complete-complete matching. Since it exploits the local congruity of features, the algorithm may easily be applied to complete-partial matching and partial-partial matching.

The transformations are computed based on (local) subsets of vertices. If there exist one or more parts that have a similar counterpart, the transformations computed for these parts will form clusters that yield candidate transformations.

As described for the complete-complete matching, for S_1 and S_2 being sets of polylines, a vertex $p_{1,0}$ of a polyline $P_1 \in S_1$ and a vertex $p_{2,0}$ of a polyline $P_2 \in S_2$ are randomly chosen in every step. Beginning with these seed vertices an ordered set $\tilde{P}_1 \subseteq P_1$ and an ordered set $\tilde{P}_2 \subseteq P_2$ of vertices or vertex surrogates are generated.

For both polylines starting from the last vertex added to \tilde{P} the distance to the next vertex is computed. If the difference of the distances lies below a certain threshold, both vertices are added, otherwise for the vertex with smaller distance a corresponding vertex surrogate with the same distance is created on the other polyline and this pair is added. For every cardinality \tilde{m} of the \tilde{P} 's the transformation t is computed, such that $\varepsilon = \sum_{i=1}^{\tilde{m}} \|t(p_{1,i}) - p_{2,i}\|^2$ is minimized. The transformation's weight is composed of the length of the covered part of

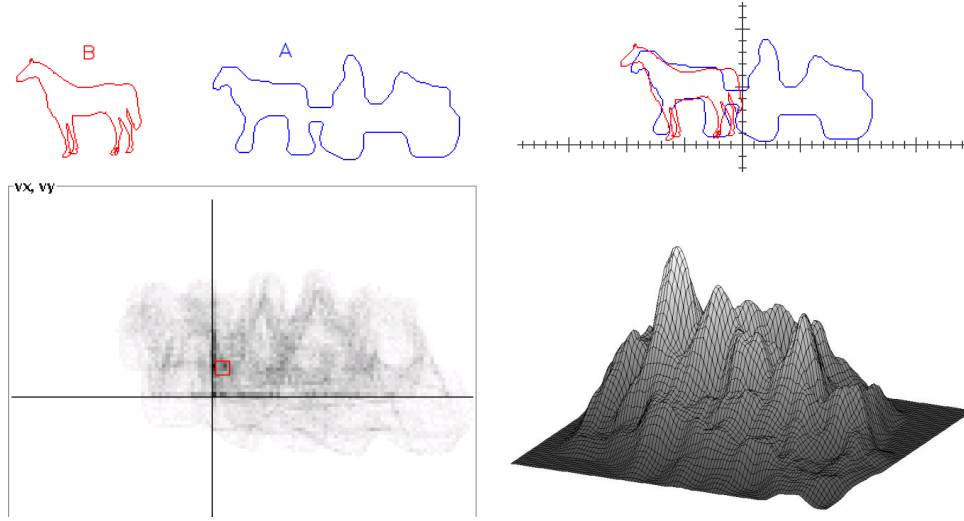


Figure 2: Top: two shapes and the superimposition found by the probabilistic algorithm. Bottom: the corresponding probability distribution in the translation plane with one significant maximum.

the polyline and the error ε . For every step the transformation t with the highest weight is handed over to the clustering algorithm.

If no complete-complete matching exists, the portions of the polylines matched will be smaller, but still the largest clusters will represent parts that do match.

2.3 Partial similarity of matched sets of line segments or polygonal curves

There is no need to change the resemblance function as described in report 4.1 - only the appraisal has to be adjusted.

The resemblance function ϕ_s for a line s is defined as

$$\phi_s(\lambda) = \max_{g \in S_2} (\alpha_{s,g}(\lambda) \cdot \beta_{s,g})$$

with α being the distance factor and β being the slope factor as defined for the complete-complete matching case.

The weighting function ω (to prevent parts with many parallel lines from dominating over parts with solitary lines) is defined analogical to the resemblance function:

$$\omega_s(\lambda) = \frac{1}{\sum_{g \in S_1} (\alpha_{s,g}(\lambda) \cdot \beta_{s,g})}$$

The directed resemblance measure $\Phi_{\rightarrow}(S_1, S_2)$ is defined by

$$\Phi_{\rightarrow}(S_1, S_2) = \frac{\sum_{s \in S_1} \left(\int_{\lambda=0}^1 \phi_s(\lambda) \cdot \omega_s(\lambda) d\lambda \cdot l_s \right)}{\Omega(S_1)}$$

with l_s being the length of s and $\Omega(S_1)$ being the total weight of S_1 :

$$\Omega(S_1) = \sum_{s \in S_1} \left(\int_{\lambda=0}^1 \omega_s(\lambda) d\lambda \cdot l_s \right)$$

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

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

The value of the undirected resemblance measure for two shapes that have a slight similarity – for the whole shape – may be as large as the value of the resemblance measure for two shapes that have parts resembling each other as well as parts that differ much from each other. Depending on a threshold for the value of ϕ , these parts may be distinguished and the resemblance measure computed for the parts with high weight. The more subtle task is to decide whether these parts form an expressive subshape that has been recognized – a match in the PPM / CPM sense – or if they are just independent and do not induce a perceptually relevant resemblance.

2.4 Turning function based matching

We have introduced a measure for computing the similarity between multiple polylines and a polygon, see figure 3. The polylines could for example be pieces of an object contour incompletely extracted from an image, or could be boundary parts in a decomposition of an object contour. The measure we propose is based on the turning function representation of the polylines and the polygon. This similarity measure is a turning angle function-based similarity, minimized over all possible shiftings of the endpoints of the parts over the shape, and also over all independent rotations of the parts. Since we allow the parts to rotate independently, this measure could capture the similarity between contours of non-rigid objects, with parts in different relative positions. We then derive a number of non-trivial properties of the similarity measure.

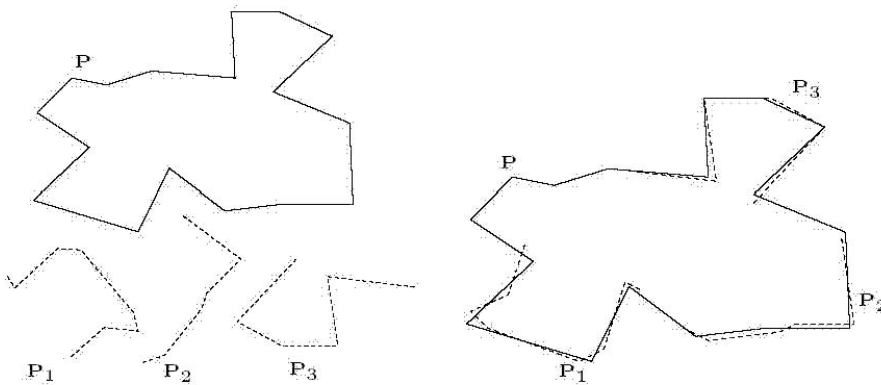


Figure 3: Matching an ordered set $\{P_1, P_2, P_3\}$ of polylines against a polygon P .

Based on these properties we characterize the optimal solution that leads to a straightforward $O(km^2n^2)$ -time and space dynamic programming algorithm using $O(km^2n^2)$ storage. We then present a novel $O(kmn \log(mn))$ time and space algorithm using $O(kmn \log(mn))$ storage. Here, m denotes the number of vertices in the polygon, and n is the total number

of vertices in the k polylines that are matched against the polygon. For more details see Appendix A Multiple Polyline to Polygon Matching.

We have experimented with a part-based retrieval application. Given a large collection of shapes and a query consisting of a set of polylines, we want to retrieve those shapes in the collection that best match our query. The set of polylines forming the query are boundary parts in a decomposition of a database shape – both this database shape and the parts in the query are selected by the user. The evaluation using a known ground-truth indicates that a part-based approach improves the global matching performance for difficult categories of shapes. For more details, see Appendix B Part-based Shape Retrieval.

3 Conclusions

Our experiments showed that the probabilistic matching algorithms can be applied to the problem of partial matching with satisfactory results. Nevertheless, we believe that some modifications of the algorithms could be quite helpful to meet the needs of the partial matching problem. Therefore, we decided to add such modifications to the implementation we are currently working on.

On the one hand, we are modifying the matching algorithms to record which parts of the given shapes are actually matched by the resulting transformation. On the other hand, the valuation function, which returns a value indicating the quality of match, could be modified in order to weight stronger the parts of the two shapes that match, and to ignore or give a smaller weight to the unmatched parts. There is a certain trade-off between the quality of the match and the size of the matched parts, which has to be reflected by the valuation function. In general, depending on the algorithm parameters, we can have a precise match of small parts, or a more rough match of the larger parts of the shapes. Which outcome is desired depends highly on the application and is therefore best controlled by the user. Similarly, it is a difficult task to decide to which extent the unmatched parts should affect the valuation function. We continue the experimental evaluation in order to find a good combination for the valuation function for the problem of trademark image matching.

We are also working on determining automatically the best parameter value δ controlling the trade-off between the quality and the size of match. For this purpose we analyze the valuation function as a function of parameter δ .

After having implemented these ideas, we expect to obtain reasonable results for the application of recognizing similarities in trademarks. Further experiments are planned for the fine-tuning the methods for this particular application.

References

- [1] Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13:251–265, 1995.
- [2] M. Hagedoorn and R. Veltkamp. State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.
- [3] Remco C. Veltkamp. Shape matching: Similarity measures and algorithms. Technical Report UU-CS-2001-03, Utrecht University, 2001.

Multiple Polyline to Polygon Matching

Mirela Tănase¹, Remco C. Veltkamp¹, and Herman Haverkort²

¹ Department of Computing Science, Utrecht University, The Netherlands

² Department of Computing Science, TU Eindhoven, The Netherlands

Abstract. We introduce a measure for computing the similarity between multiple polylines and a polygon, that can be computed in $O(km^2n^2)$ time with a straightforward dynamic programming algorithm. We then present a novel fast algorithm that runs in time $O(kmn \log mn)$. Here, m denotes the number of vertices in the polygon, and n is the total number of vertices in the k polylines that are matched against the polygon. The effectiveness of the similarity measure has been demonstrated in a part-based retrieval application with known ground-truth.

1 Introduction

The motivation for multiple polyline to polygon matching is twofold. Firstly, the matching of shapes has been done mostly by comparing them as a whole [2,8,10]. This fails when a significant part of one shape is occluded, or distorted by noise. In this paper, we address the partial shape matching problem, matching portions of two given shapes. Secondly, partial matching helps identifying similarities even when a significant portion of one shape boundary is occluded, or seriously distorted. It could also help in identifying similarities between contours of a non-rigid object in different configurations of its moving parts, like the contours of a sitting and a walking cat. Finally, partial matching helps alleviating the problem of unreliable object segmentation from images, over or undersegmentation, giving only partially correct contours.

Contribution. Firstly, we introduce a measure for computing the similarity between multiple polylines and a polygon. This similarity measure is a turning angle function-based similarity, minimized over all possible shiftings of the endpoints of the parts over the shape, and also over all independent rotations of the parts. Since we allow the parts to rotate independently, this measure could capture the similarity between contours of non-rigid objects, with parts in different relative positions. We then derive a number of non-trivial properties of the similarity measure.

Secondly, based on these properties we characterize the optimal solution that leads to a straightforward $O(km^2n^2)$ -time and space dynamic programming algorithm. We then present a novel $O(kmn \log mn)$ time and space algorithm. Here, m denotes the number of vertices in the polygon, and n is the total number of vertices in the k polylines that are matched against the polygon.

Thirdly, we have experimented with a part-based retrieval application. Given a large collection of shapes and a query consisting of a set of polylines, we want to

retrieve those shapes in the collection that best match our query. The evaluation using a known ground-truth indicates that a part-based approach improves the global matching performance for difficult categories of shapes.

2 Related Work

Arkin et al. [2] describe a metric for comparing two whole polygons that is invariant under translation, rotation and scaling. It is based on the L_2 -distance between the turning functions of the two polygons, and can be computed in $O(mn \log mn)$ time, where m is the number of vertices in one polygon and n is the number of vertices in the other.

Most partial shape matching methods are based on computing local features of the contour, and then looking for correspondences between the features of the two shapes, for example points of high curvature [1,7]. Such local features-based solutions work well when the matched subparts are almost equivalent up to a transformation such as translation, rotation or scaling, because for such subparts the sequences of local features are very similar. However, parts that we perceive as similar, may have quite different local features (different number of curvature extrema for example).

Geometric hashing [12] is a method that determines if there is a transformed subset of the query point set that matches a subset of a target point set, by building a hash table in transformation space. Also the Hausdorff distance [5] allows partial matching. It is defined for arbitrary non-empty bounded and closed sets A and B as the infimum of the distance of the points in A to B and the points in B to A . Both methods are designed for partial matching, but do not easily transform to our case of matching multiple polylines to a polygon.

Partially matching the turning angle function of two polylines under scaling, translation and rotation, can be done in time $O(m^2n^2)$ [4]. Given two matches with the same squared error, the match involving the longer part of the polylines has a lower dissimilarity. The dissimilarity measure is a function of the scale, rotation, and the shift of one polyline along the other. However, this works for only two single polylines.

Latecki et al [6], establish the best correspondence of parts in a decomposition of the matched shapes. The best correspondence between the maximal convex arcs of two simplified versions of the original shapes gives the partial similarity measure between the shapes. One drawback of this approach is that the matching is done between parts of simplified shapes at “the appropriate evolution stage”. How these evolution stages are identified is not indicated in their papers, though it certainly has an effect on the quality of the matching process.

3 Polylines-to-Polygon Matching

We concentrate on the problem of matching an ordered set $\{P_1, P_2, \dots, P_k\}$ of k polylines against a polygon P . We want to compute how close an ordered set of polylines $\{P_1, P_2, \dots, P_k\}$ is to being part of the boundary of P in the given

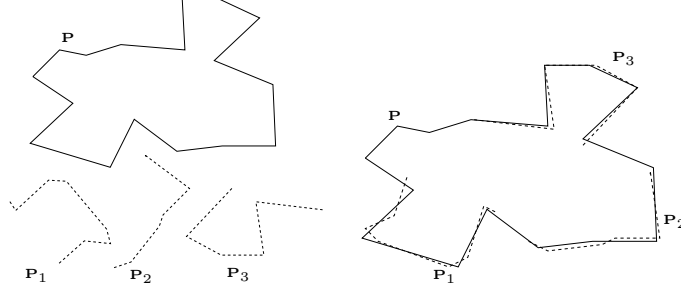


Fig. 1. Matching an ordered set $\{P_1, P_2, P_3\}$ of polylines against a polygon P

order in counter-clockwise direction around P (see figure 1). For this purpose, the polylines are rotated and shifted along the polygon P , in such a way that the pieces of the boundary of P “covered” by the k polylines are mutually disjoint except possibly at their endpoints. Note that P is a polygon and not an open polyline, only because of the intended application of part based retrieval.

3.1 Similarity Between Multiple Polylines and a Polygon

The *turning function* Θ_A of a polygon A measures the angle of the counter-clockwise tangent with respect to a reference orientation as a function of the arc-length s , measured from some reference point on the boundary of A . It is a piecewise constant function, with jumps corresponding to the vertices of A . A rotation of A by an angle θ corresponds to a shifting of Θ_A over a distance θ in the vertical direction. Moving the location of the reference point $A(0)$ over a distance $t \in [0, l_A]$ along the boundary of A corresponds to shifting Θ_A horizontally over a distance t .

Let $\Theta : [0, l] \rightarrow \mathbb{R}$ be the turning function of a polygon P of m vertices, and of perimeter length l . Since P is a closed polyline, the domain of Θ can be easily extended to the entire real line, by $\Theta(s + l) = \Theta(s) + 2\pi$. Let $\{P_1, P_2, \dots, P_k\}$ be a set of polylines, and let $\Theta_j : [0, l_j] \rightarrow \mathbb{R}$ denote the turning function of the polyline P_j of length l_j . If P_j is made of n_j segments, Θ_j is piecewise-constant with $n_j - 1$ jumps.

For simplicity of exposition, $f_j(t, \theta)$ denotes the quadratic similarity between the polyline P_j and the polygon P , for a given placement (t, θ) of P_j over P : $f_j(t, \theta) = \int_0^{l_j} (\Theta(s + t) - \Theta_j(s) + \theta)^2 ds$.

We assume the polylines $\{P_1, P_2, \dots, P_k\}$ satisfy the condition $\sum_{j=1}^k l_j \leq l$. The similarity measure, which we denote by $d(P_1, \dots, P_k; P)$, is the square root of the sum of quadratic similarities f_j , minimized over all valid placements of P_1, \dots, P_k over P :

$$d(P_1, \dots, P_k; P) = \min_{\substack{\text{valid placements} \\ (t_1, \theta_1) \dots (t_k, \theta_k)}} \left(\sum_{j=1}^k f_j(t_j, \theta_j) \right)^{1/2}.$$

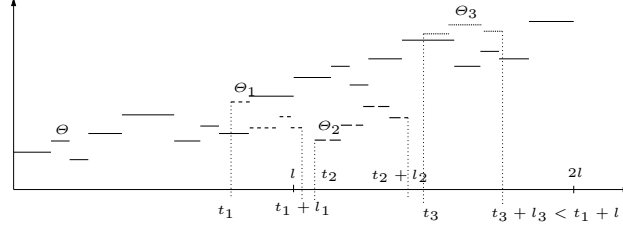


Fig. 2. To compute $d(P_1, \dots, P_k; P)$ between the polylines P_1, \dots, P_k and the polygon P , we shift the turning functions Θ_1, Θ_2 , and Θ_3 horizontally and vertically over Θ

It remains to define what the valid placements are. The horizontal shifts t_1, \dots, t_k correspond to shiftings of the starting points of the polylines P_1, \dots, P_k along P . We require that the starting points of P_1, \dots, P_k are matched with points on the boundary of P in counterclockwise order around P , that is: $t_{j-1} \leq t_j$ for all $1 < j \leq k$, and $t_k \leq t_1 + l$. Furthermore, we require that the matched parts are disjoint (except possibly at their endpoints), sharpening the constraints to $t_{j-1} + l_{j-1} \leq t_j$ for all $1 < j \leq k$, and $t_k + l_k \leq t_1 + l$ (see figure 2).

The vertical shifts $\theta_1, \dots, \theta_k$ correspond to rotations of the polylines P_1, \dots, P_k with respect to the reference orientation, and are independent of each other. Therefore, in an optimal placement the quadratic similarity between a particular polyline P_j and P depends only on the horizontal shift t_j , while the vertical shift must be optimal for the given horizontal shift. We can thus express the similarity between P_j and P for a given positioning t_j of the starting point of P_j over P as: $f_j^*(t_j) = \min_{\theta \in \mathbb{R}} f_j(t_j, \theta)$.

The similarity between the polylines P_1, \dots, P_k and the polygon P is thus:

$$d(P_1, \dots, P_k; P) = \min_{\substack{t_1 \in [0, l], t_2, \dots, t_k \in [0, 2l]; \\ \forall j \in \{2, \dots, k\}: t_{j-1} + l_{j-1} \leq t_j; \quad t_k + l_k \leq t_1 + l}} \left(\sum_{j=1}^k f_j^*(t_j) \right)^{1/2}. \quad (1)$$

3.2 Properties of the Similarity Function

In this section we give a few properties of $f_j^*(t)$, as functions of t , that constitute the basis of the algorithms for computing $d(P_1, \dots, P_k; P)$ in sections 3.3 and 3.4. We also give a simpler formulation of the optimization problem in the definition of $d(P_1, \dots, P_k; P)$. Arkin et al. [2] have shown that for any fixed t , the function $f_j(t, \theta)$ is a quadratic convex function of θ . This implies that for a given t , the optimization problem $\min_{\theta \in \mathbb{R}} f_j(t, \theta)$ has a unique solution, given by the root $\theta_j^*(t)$ of the equation $\partial f_j(t, \theta) / \partial \theta = 0$. As a result:

Lemma 1. For a given positioning t of the starting point of P_j over P , the rotation that minimizes the quadratic similarity between P_j and P is given by $\theta_j^*(t) = -\int_0^{l_j} (\Theta(s+t) - \Theta_j(s)) ds / l_j$.

We now consider the properties of $f_j^*(t) = f_j(t, \theta_j^*(t))$, as a function of t .

Lemma 2. *The quadratic similarity $f_j^*(t)$ has the following properties:*

- i) *it is periodic, with period l ;*
- ii) *it is piecewise quadratic, with mn_j breakpoints within any interval of length l ; moreover, the parabolic pieces are concave.*

For a proof of this and the following lemmas, see [11].

The following corollary indicates that to compute the minimum of the function f_j^* , we need to look only a discrete set of at most mn_j points.

Corollary 1. *The local minima of the function f_j^* are among the breakpoints between its parabolic pieces.*

We now give a simpler formulation of the optimization problem in the definition of $d(P_1, \dots, P_k; P)$. In order to simplify the restrictions on t_j in equation (1), we define: $\bar{f}_j(t) := f_j^*(t + \sum_{i=1}^{j-1} l_i)$. In other words, the function \bar{f}_j is a copy of f_j^* , but shifted to the left with $\sum_{i=1}^{j-1} l_i$. Obviously, \bar{f}_j has the same properties as f_j^* , that is: it is a piecewise quadratic function of t that has its local minima in at most mn_j breakpoints in any interval of length l . With this simple transformation of the set of functions f_j^* , the optimization problem defining $d(P_1, \dots, P_k; P)$ becomes:

$$d(P_1, \dots, P_k; P) = \min_{\substack{t_1 \in [0, l), t_2, \dots, t_k \in [0, 2l); \\ \forall j \in \{2, \dots, k\}: t_{j-1} \leq t_j; \quad t_k \leq t_1 + l_0}} \left(\sum_{j=1}^k \bar{f}_j(t_j) \right)^{1/2}, \quad (2)$$

where $l_0 := l - \sum_{i=1}^k l_i$. Notice that if $(\bar{t}_1^*, \dots, \bar{t}_k^*)$ is a solution to the optimization problem in equation (2), then (t_1^*, \dots, t_k^*) , with $t_j^* := \bar{t}_j^* + \sum_{i=1}^{j-1} l_i$, is a solution to the optimization problem in equation (1).

3.3 Characterization of an Optimal Solution

In this section we characterize the structure of an optimal solution to the optimization problem in equation (2), and give a recursive definition of this solution. This definition forms the basis of a straightforward dynamic programming solution to the problem.

Let $(\bar{t}_1^*, \dots, \bar{t}_k^*)$ be a solution to the optimization problem in equation (2).

Lemma 3. *The values of an optimal solution $(\bar{t}_1^*, \dots, \bar{t}_k^*)$ are found in a discrete set of points $X \subset [0, 2l)$ of the breakpoints of the functions $\bar{f}_1, \dots, \bar{f}_k$, plus two copies of each breakpoint: one shifted left by l_0 and one shifted right by l_0 .*

We call a point in $[0, 2l)$, which is either a breakpoint of $\bar{f}_1, \dots, \bar{f}_k$, or such a breakpoint shifted left or right by l_0 , a *critical point*. Since function \bar{f}_j has $2mn_j$ breakpoints, the total number of critical points in $[0, 2l)$ is at most $6m \sum_{i=1}^k n_i = 6mn$. Let $X = \{x_0, \dots, x_{N-1}\}$ be the set of critical points in $[0, 2l)$.

With the observations above, the optimization problem we have to solve is:

$$d(P_1, \dots, P_k; P) = \min_{\substack{t_1, \dots, t_k \in X \\ \forall j > 1 : t_{j-1} \leq t_j; t_k - t_1 \leq l_0}} \left(\sum_{j=1}^k \bar{f}_j(t_j) \right)^{1/2}. \quad (3)$$

We denote:

$$D[j, a, b] = \min_{\substack{t_1, \dots, t_j \in X \\ x_a \leq t_1 \leq \dots \leq t_j \leq x_b}} \sum_{i=1}^j \bar{f}_i(t_i), \quad (4)$$

where $j \in \{1, \dots, k\}$, $a, b \in \{0, \dots, N-1\}$, and $a \leq b$. Equation (4) describes the subproblem of matching the set $\{P_1, \dots, P_j\}$ of j polylines to a subchain of P , starting at $P(x_a)$ and ending at $P(x_b + \sum_{i=1}^j l_i)$. We now show that $D[j, a, b]$ can be computed recursively. Let $(t_1^{\otimes}, \dots, t_j^{\otimes})$ be an optimal solution for $D[j, a, b]$. Regarding the value of t_j^{\otimes} we distinguish two cases:

- $t_j^{\otimes} = x_b$, in which case $(t_1^{\otimes}, \dots, t_{j-1}^{\otimes})$ must be an optimal solution for $D[j-1, a, b]$, otherwise $(t_1^{\otimes}, \dots, t_j^{\otimes})$ would not give a minimum for $D[j, a, b]$; thus in this case, $D[j, a, b] = D[j-1, a, b] + \bar{f}_j(x_b)$;
- $t_j^{\otimes} \neq x_b$, in which case $(t_1^{\otimes}, \dots, t_j^{\otimes})$ must be an optimal solution for $D[j, a, b-1]$; otherwise $(t_1^{\otimes}, \dots, t_j^{\otimes})$ would not give a minimum for $D[j, a, b]$; thus in this case $D[j, a, b] = D[j, a, b-1]$.

We can now conclude that :

$$D[j, a, b] = \min (D[j-1, a, b] + \bar{f}_j(x_b), D[j, a, b-1]), \text{ for } j \geq 1 \wedge a \leq b, \quad (5)$$

where the boundary cases are $D[0, a, b] = 0$ and $D[j, a, a-1]$ has no solution.

A solution of the optimization problem (3) is then given by

$$d(P_1, \dots, P_k; P) = \min_{x_a, x_b \in X, x_b - x_a \leq l_0} \sqrt{D[k, a, b]}. \quad (6)$$

Equations (5) and (6) lead to a straightforward dynamic programming algorithm for computing the similarity measure $d(P_1, \dots, P_k; P)$ in $O(km^2n^2)$ time.

3.4 A Fast Algorithm

The above time bound to compute of the similarity measure $d(P_1, \dots, P_k; P)$ can be improved to $O(kmn \log mn)$. The refinement of the dynamic programming algorithm is based on the following property of equation (5):

Lemma 4. *For any polyline P_j , $j \in \{1, \dots, k\}$, and any critical point x_b , $b \in \{0, \dots, N-1\}$, there is a critical point x_z , $0 \leq z \leq b$, such that:*

- i) $D[j, a, b] = D[j, a, b-1]$, for all $a \in \{0, \dots, z-1\}$, and
- ii) $D[j, a, b] = D[j-1, a, b] + \bar{f}_j(x_b)$, for all $a \in \{z, \dots, b\}$.

For given j and b , we consider the function $\mathcal{D}[j, b] : \{0, N-1\} \rightarrow \mathbb{R}$, with $\mathcal{D}[j, b](a) = D[j, a, b]$. Lemma 4 expresses the fact that the values of function $\mathcal{D}[j, b]$ can be obtained from $\mathcal{D}[j, b-1]$ up to some value z , and from $\mathcal{D}[j-1, b]$ (while adding $\bar{f}_j(x_b)$) from this value onwards. This property allows us to improve the time bound of the dynamic programming algorithm. Instead of computing arrays of scalars $D[j, a, b]$, we will compute arrays of functions $\mathcal{D}[j, b]$. The key to success will be to represent these functions in such a way that they can be evaluated fast and $\mathcal{D}[j, b]$ can be constructed from $\mathcal{D}[j-1, b]$ and $\mathcal{D}[j, b-1]$ fast.

Algorithm FastCompute $\mathbf{d}(P_1, \dots, P_k; P)$

1. Compute the set of critical points $X = \{x_0, \dots, x_{N-1}\}$, and sort them
2. For all $j \in \{1, \dots, k\}$ and all $b \in \{0, \dots, N-1\}$, evaluate $\bar{f}_j(x_b)$
3. *ZERO* \leftarrow a function that always evaluates to zero (see Lemma 5)
4. *INFINITY* \leftarrow a function that always evaluates to ∞ (see Lemma 5)
5. *MIN* $\leftarrow \infty$
6. $a \leftarrow 0$
7. **for** $j \leftarrow 1$ **to** k **do**
8. $\mathcal{D}[j, -1] \leftarrow$ *INFINITY*
9. **for** $b \leftarrow 0$ **to** $N-1$ **do**
10. $\mathcal{D}[0, b] \leftarrow$ *ZERO*
11. **for** $j \leftarrow 1$ **to** k **do**
12. Construct $\mathcal{D}[j, b]$ from $\mathcal{D}[j-1, b]$ and $\mathcal{D}[j, b-1]$
13. **while** $x_a < x_b - l_0$ **do**
14. $a \leftarrow a + 1$
15. val \leftarrow evaluation of $\mathcal{D}[k, b](a)$
16. *MIN* $\leftarrow \min(\text{val}, \text{MIN})$
17. **return** $\sqrt{\text{MIN}}$

The running time of this algorithm depends on how the functions $\mathcal{D}[j, b]$ are represented. In order to make especially steps 12 and 15 of the above algorithm efficient, we represent the functions $\mathcal{D}[j, b]$ by means of balanced binary trees. Asano et al. [3] used an idea similar in spirit.

An efficient representation for function $\mathcal{D}[j, b]$. We now describe the tree $\mathcal{T}_{j,b}$ used for storing the function $\mathcal{D}[j, b]$. Each node ν of $\mathcal{T}_{j,b}$ is associated with an interval $[a_\nu^-, a_\nu^+]$, with $0 \leq a_\nu^- \leq a_\nu^+ \leq N-1$. The root ρ is associated with the full domain, that is: $a_\rho^- = 0$ and $a_\rho^+ = N-1$. Each node ν with $a_\nu^- < a_\nu^+$ is an internal node that has a split value $a_\nu = \lfloor (a_\nu^- + a_\nu^+) / 2 \rfloor$ associated with it. Its left and right children are associated with $[a_\nu^-, a_\nu]$ and $[a_\nu + 1, a_\nu^+]$, respectively. Each node ν with $a_\nu^- = a_\nu^+$ is a leaf of the tree, with $a_\nu = a_\nu^- = a_\nu^+$. For any index a of a critical point x_a , we will denote the leaf ν that has $a_\nu = a$ by λ_a . Note that so far, the tree looks exactly the same for each function $\mathcal{D}[j, b]$: they are balanced binary trees with N leaves, and $\log N$ height. Moreover, all trees have the same associated intervals, and split values in their corresponding nodes. With each node ν we also store a weight w_ν , such that $\mathcal{T}_{j,b}$ has the following property: $\mathcal{D}[j, b](a)$ is the sum of the weights on the path from the tree root to the leaf λ_a . Such a representation of a function $\mathcal{D}[j, b]$ is not unique. Furthermore, we store with each node ν a value m_ν which is the sum of the weights on the

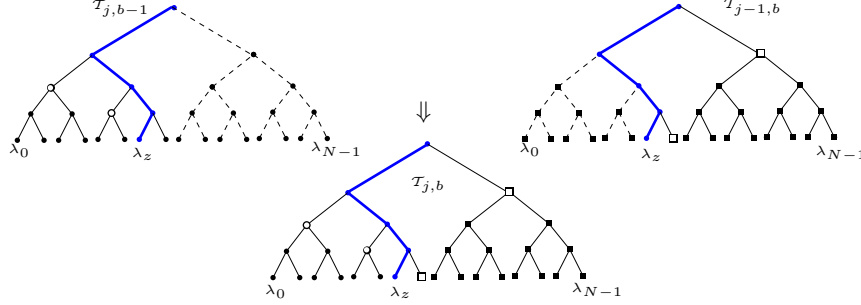


Fig. 3. The tree $\mathcal{T}_{j,b}$ is constructed from $\mathcal{T}_{j,b-1}$ and $\mathcal{T}_{j-1,b}$ by creating new nodes along the path from the root to the leaf λ_z , and adopting the subtrees to the left of the path from $\mathcal{T}_{j,b-1}$, and the subtrees to the right of the path from $\mathcal{T}_{j-1,b}$

path from the left child of ν to the leaf $\lambda_{a\nu}$, that is: the rightmost descendant of the left child of ν .

Lemma 5. *The data structure $\mathcal{T}_{j,b}$ for the representation of function $\mathcal{D}[j, b]$ can be operated on such that:*

- (i) *The representation of a zero-function (i.e. a function that always evaluates to zero) can be constructed in $O(N)$ time. Also the representation of a function that always evaluates to ∞ can be constructed in $O(N)$ time.*
- (ii) *Given $\mathcal{T}_{j,b}$ of $\mathcal{D}[j, b]$, evaluating function $\mathcal{D}[j, b](a)$ takes $O(\log N)$ time.*
- (iii) *Given $\mathcal{T}_{j-1,b}$ and $\mathcal{T}_{j,b-1}$ of the functions $\mathcal{D}[j-1, b]$ and $\mathcal{D}[j, b-1]$, respectively, a representation $\mathcal{T}_{j,b}$ of $\mathcal{D}[j, b]$ can be computed in $O(\log N)$ time.*

Proof. We restrict ourselves to item (iii), the main element of the solution.

To construct $\mathcal{T}_{j,b}$ from $\mathcal{T}_{j,b-1}$ and $\mathcal{T}_{j-1,b}$ efficiently, we take the following approach. We find the sequences of left and right turns that lead from the root of the trees down to the leaf λ_z , where z is defined as in lemma 4. Note that the sequences of left and right turns are the same in the trees $\mathcal{T}_{j,b}$, $\mathcal{T}_{j,b-1}$, and $\mathcal{T}_{j-1,b}$, only the weights on the path differ. Though we do not compute z explicitly, we will show below that we are able to construct the path from the root of the tree to the leaf λ_z corresponding to z , by identifying, based on the stored weights, at each node along this path whether the path continues left or right.

Lemma 4 tells us that for each leaf left of λ_z , the total weight on the path to the root in $\mathcal{T}_{j,b}$ must be the same as the total weight on the corresponding path in $\mathcal{T}_{j,b-1}$. At λ_z itself and right of λ_z , the total weights to the root in $\mathcal{T}_{j,b}$ must equal those in $\mathcal{T}_{j-1,b}$, plus $\bar{f}_j(x_z)$. We construct the tree $\mathcal{T}_{j,b}$ with these properties as follows. We start building $\mathcal{T}_{j,b}$ by constructing a root ρ . If the path to λ_z goes into the right subtree, we adopt as left child of ρ the corresponding left child ν of the root from $\mathcal{T}_{j,b-1}$. There is no need to copy ν : we just add a pointer to it. Furthermore, we set the weight of ρ equal to the weight of the root of $\mathcal{T}_{j,b-1}$. If the path to λ_z goes into the left subtree, we adopt the right child from $\mathcal{T}_{j-1,b}$ and take the weight of ρ from there, now adding $\bar{f}_j(x_z)$.

Then we make a new root for the other subtree of the root ρ , i.e. the one that contains λ_z , and continue the construction process in that subtree. Every time we go into the left branch, we adopt the right child from $\mathcal{T}_{j-1,b}$, and every time we go into the right branch, we adopt the left child from $\mathcal{T}_{j,b-1}$ (see figure 3). For every constructed node ν , we set its weight w_ν so that the total weight of ν and its ancestors equals the total weight of the corresponding nodes in the tree from which we adopt ν 's child — if the subtree adopted comes from $\mathcal{T}_{j-1,b}$, we increase w_ν by $\bar{f}_j(x_z)$.

By keeping track of the accumulated weights on the path down from the root in all the trees, we can set the weight of each newly constructed node ν correctly in constant time per node. The accumulated weights together with the stored weights for the paths down to left childrens' rightmost descendants, also allow us to decide in constant time which is better: $\mathcal{D}[j, b-1](a_\nu)$ or $\mathcal{D}[j-1, b](a_\nu) + \bar{f}_j(x_z)$. This will tell us if λ_z is to be found in the left or in the right subtree of ν .

The complete construction process only takes $O(1)$ time for each node on the path from ρ to λ_z . Since the trees are perfectly balanced, this path has only $O(\log N)$ nodes, so that $\mathcal{T}_{j,b}$ is constructed in time $O(\log N)$. \square

Theorem 1. *The similarity $d(P_1, \dots, P_k; P)$ between k polylines $\{P_1, \dots, P_j\}$ with n vertices in total, and a polygon P with m vertices, can be computed in $O(kmn \log(mn))$ time using $O(kmn \log(mn))$ storage.*

Proof. We use algorithm Fastcompute $d(P_1, \dots, P_k; P)$ with the data structure described above. Step 1 and 2 of the algorithm can be executed in $O(kmn + mn \log n)$ time. From lemma 5, we have that the zero-function *ZERO* can be constructed in $O(N)$ time (line 3). Similarly, the infinity-function *INFINITY* can be constructed in $O(N)$ time (line 4). Lemma 5 also insures that constructing $\mathcal{D}[j, b]$ from $\mathcal{D}[j-1, b]$ and $\mathcal{D}[j, b-1]$ (line 12) takes $O(\log N)$ time, and that the evaluation of $\mathcal{D}[k, b](a)$ (line 15) takes $O(\log N)$ time. Notice that no node is ever edited after it has been constructed. Thus, the total running time of the above algorithm will be dominated by $O(kN)$ executions of line 12, taking in total $O(kN \log N) = O(kmn \log(mn))$ time.

Apart from the function values of \bar{f}_j computed in step 2, we have to store the *ZERO* and the *INFINITY* function. All these require $O(kN) = O(kmn)$ storage. Notice that any of the functions constructed in step 12 requires only storing $O(\log(N)) = O(\log(mn))$ new nodes and pointers to nodes in previously computed trees, and thus we need $O(kmn \log(mn))$ for all the trees computed in step 12. So the total storage required by the algorithm is $O(kmn \log(mn))$. \square

We note that the problem resembles a general edit distance type approximate string matching [9]. Global string matching under a general edit distance error model can be done by dynamic programming in $O(kN)$ time, where k and N represent the lengths of the two strings. The same time complexity can be achieved for partial string matching through a standard “assign first line to zero” trick [9]. This however does not apply here due to the condition $x_b - x_a \leq l_0$.

4 Experimental Results

Our algorithm has been implemented in C++ and is evaluated in a part-based shape retrieval application (see <http://give-lab.cs.uu.nl/Matching/Mtam/>) with the Core Experiment “CE-Shape-1” part B test set devised by the MPEG-7 group to measure the performance of similarity-based retrieval for shape descriptors. This test set consists of 1400 images: 70 shape classes, with 20 images per class. The shape descriptor selected by MPEG-7 to represent a closed contour of a 2D object or region in an image is based on the Curvature Scale Space (CSS) representation [8]. We compared our matching to the CSS method, as well as to matching the global contours with turning angle functions (GTA).

The performance of each shape descriptor was measured using the “bull’s-eye” percentage: the percentage of retrieved images belonging to the same class among the top 40 matches (twice the class size). These experimental results indicate that for those classes with a low performance of the CSS matching, our approach consistently performs better. See figure 4 for two examples. The emphasis of this paper lies on the algorithmic aspects, but for a rigorous experimental evaluation, see [11]. The running time for a single query on the MPEG-7 test set of 1400 images is typically about one second on a 2 GHz PC.




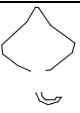
CSS/GTA Query Image	Part-based Query Parts	Bull’s Eye Score		
		CSS	GTA	MPP
 beetle-20		10	30	65
 ray-3		15	15	70

Fig. 4. A comparison of the Curvature Scale Space (CSS), Global Turning Angle function (GTA), and our Multiple Polyline to Polygon (MPP) matching (in %)

Acknowledgements. We thank Veli Mäkinen for the partial string matching reference, and Geert-Jan Giezeman for programming support. This research was supported by the FP6 IST projects 511572-2 PROF1 and 506766 AIM@SHAPE, and the Dutch Science Foundation (NWO) project 612.061.006 MINDSHADE.

References

1. N. Ansari and E. J. Delp. Partial shape recognition: A landmark-based approach. *PAMI*, 12:470–483, 1990.
2. E. Arkin, L. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell. An efficiently computable metric for comparing polygonal shapes. *PAMI*, 13:209–215, 1991.
3. T. Asano, M. de Berg, O. Cheong, H. Everett, H.J. Haverkort, N. Katoh, and A. Wolff. Optimal spanners for axis-aligned rectangles. *CGTA*, 30(1):59–77, 2005.
4. Scott D. Cohen and Leonidas J. Guibas. Partial matching of planar polylines under similarity transformations. In *Proc. SODA*, pages 777–786, 1997.

5. D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *PAMI*, 15:850–863, 1993.
6. L. J. Latecki, R. Lakämper, and D. Wolter. Shape similarity and visual parts. In *Proc. Int. Conf. Discrete Geometry for Computer Imagery*, pages 34–51, 2003.
7. H.-C. Liu and M. D. Srinath. Partial shape classification using contour matching in distance transformation. *PAMI*, 12(11):1072–1079, 1990.
8. F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Workshop on Image DataBases and MultiMedia Search*, pages 35–42, 1996.
9. Gonzala Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.
10. K. Siddiqi, A. Shokoufandeh, S.J. Dickinson, and S.W. Zucker. Shock graphs and shape matching. *IJCV*, 55(1):13–32, 1999.
11. Mirela Tanase. *Shape Deomposition and Retrieval*. PhD thesis, Utrecht University, Department of Computer Science, February 2005.
12. Haim Wolfson and Isidore Rigoutsos. Geometric hashing: an overview. *IEEE Computational Science & Engineering*, pages 10–21, October-December 1997.

Part-based Shape Retrieval

Mirela Tanase
 Department of Computer Science
 Utrecht University, The Netherlands
 mirela@cs.uu.nl

Remco C. Veltkamp
 Department of Computer Science
 Utrecht University, The Netherlands
 Remco.Veltkamp@cs.uu.nl

ABSTRACT

This paper introduces a measure for computing the dissimilarity between multiple polylines and a polygon based on the turning function, and describes a part-based retrieval system using that dissimilarity measure. This dissimilarity can be efficiently computed in time $O(kmn \log mn)$, where m denotes the number of vertices in the polygon, and n is the total number of vertices in the k polylines that are matched against the polygon. This dissimilarity measure identifies similarities even when a significant portion of one shape is different from the other, for example because the shape is articulated, or because of occlusion or distortion. The effectiveness of the dissimilarity measure is demonstrated in a part-based shape retrieval system. Quantitative experimental verification is performed with a known ground-truth, the MPEG-7 Core Experiment test set, in a comparison with the Curvature Scale Space method, and a global turning angle function method.

Categories and Subject Descriptors: I.5 Pattern Recognition, I.3.5 Computational Geometry.

General Terms: Algorithms.

Keywords: Shape matching, retrieval.

1. INTRODUCTION

The motivation for part-based shape retrieval is twofold. Firstly, partial matching is an important problem that has not received much attention. The matching of shapes has been done mostly by comparing them as a whole [1, 9, 12]. Such a matching fails when a significant part of one shape is occluded, for example. In this paper, we address the partial shape matching problem, which is concerned with matching portions of two given shapes.

Secondly, partial matching methods are effective for database retrieval problems. Partial matching helps identifying similarities even when a significant portion of one shape boundary is occluded, or seriously distorted. It could also help in identifying similarities between contours of an articulated object in different configurations of its moving parts, like the contours of a sitting and a walking cat. Partial matching also helps alleviating the problem of unreliable object segmentation from images, over or undersegmentation, giving only partially correct contours.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'05, November 6–11, 2005, Singapore.

Copyright 2005 ACM 1-59593-044-2/05/0011 ...\$5.00.

Contribution Firstly, we introduce a measure for computing the dissimilarity between multiple polylines and a polygon, see figure 1. The polylines could for example be pieces of an object contour incompletely extracted from an image, or could be boundary parts in a decomposition of an object contour. The measure we propose is based on the turning function representation, and can be computed in $O(kmn \log mn)$ time, where m denotes the number of vertices in the polygon, and n is the total number of vertices in the k polylines that are matched against the polygon.

Secondly, we describe a part-based shape retrieval system. Given a large collection of shapes, and a query consisting of a set of polylines, we want to retrieve those shapes in the collection that best match our query. The set of polylines forming the query are boundary parts in a decomposition of a database shape – both this database shape and the parts in the query are selected by the user. The quantitative evaluation on the basis of a known ground-truth indicate that a part-based approach to matching consistently improves the global matching performance for difficult categories of shapes.

1.1 Previous Work

Most of the approaches to partial shape matching are based on computing local features of the contour, and then looking for correspondences between the features of the two shapes, see e.g. [8], [11]. These local solutions work well when the matched subparts are almost equivalent up to a transformation such as translation, rotation or scaling, because for such subparts the sequences of local features are very similar. This makes them useful for applications like detecting instances of a model shape in a cluttered scene. However, the problem of shape-based retrieval in a general database is more involved, since it requires to report matchings between subparts that we perceive as similar, but may have quite different local features (different number of curvature extrema for example).

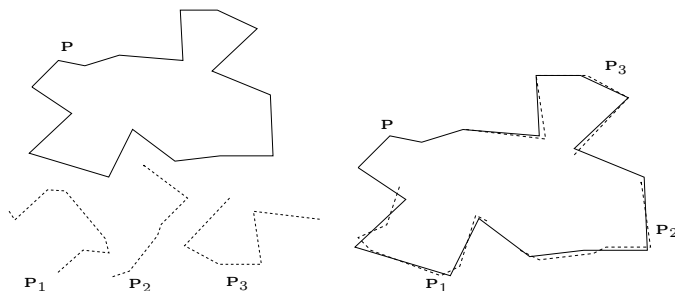


Figure 1: Matching an ordered set $\{P_1, P_2, P_3\}$ of polylines against a polygon P .

Geometric hashing [15] is designed for partial matching, but does not easily transform to our case of matching multiple polylines to a polygon. Also the Hausdorff distance allows partial matching. It is used by [4] for image matching, but only for discrete point sets.

Partial matching based on the turning function (see section 2.1) of two polylines under scaling, translation and rotation, can be done in time $O(m^2n^2)$, see [2]. However, that works for only two single polylines.

In this paper we address the partial shape matching problem for the purpose of shape-based retrieval. Another approach to this problem is that of Latecki et al. [7], which is based on establishing the best correspondence of parts in a decomposition of the matched shapes, simplified by a discrete curve evolution process. A serious drawback of this approach is that the matching is done between parts of simplified shapes at “the appropriate evolution stage”, which is not further detailed, though it certainly has an effect on the quality of the matching process.

2. POLYLINES-TO-POLYGON MATCHING

In this section we concentrate on the problem of matching an ordered set $\{P_1, P_2, \dots, P_k\}$ of k polylines against a polygon P (see figure 1). For this purpose, the polylines are rotated and shifted along the polygon P , in such a way that the pieces of the boundary of P “covered” by the k polylines are mutually disjoint except possibly at their endpoints.

2.1 Turning function

The *turning function* Θ_A of a polygon A measures the angle of the counterclockwise tangent with respect to a reference orientation as a function of the arc-length s , measured from some reference point on the boundary of A . It is a piece-wise constant function, with jumps corresponding to the vertices of A . A rotation of A by an angle θ corresponds to a shifting of Θ_A over a distance θ in the vertical direction. Moving the location of the reference point $A(0)$ over a distance $t \in [0, l_A)$ along the boundary of A corresponds to shifting Θ_A horizontally over a distance t .

The distance between *two polygons* A and B is defined as the L_2 norm between their two turning functions Θ_A and Θ_B , minimized with respect to the vertical and horizontal shifts of these functions (in other words, minimized with respect to rotation and choice of reference point). More formally, suppose A and B are two polygons with perimeter length l_A and l_B , respectively, and the polygon B is placed over A in such a way that the reference point $B(0)$ of B coincides with point $A(t)$ at distance t along A from the reference point $A(0)$, and B is rotated clockwise by an angle θ with respect to the reference orientation. We define the quadratic dissimilarity $f(A, B, t, \theta)$ between A and B for a given placement (t, θ) of B over A , as the square of the L_2 norm between their two turning functions Θ_A and Θ_B , shifted relative to each other corresponding to the values of t and θ :

$$f(A, B, t, \theta) = \int_0^{l_B} (\Theta_A(s+t) - \Theta_B(s) + \theta)^2 ds.$$

The dissimilarity between two polygons A and B is then given by:

$$d(A, B) = \min_{\theta \in \mathbb{R}, t \in [0, l_A)} \sqrt{f(A, B, t, \theta)}.$$

To achieve invariance under scaling, Arkin et al. [1] propose to normalize the two polygons to unit length prior to the matching.

For measuring the difference between a polygon A and a polyline B the same measure can be used. For the purpose of our part-based retrieval application, we want that a polyline B included in

a polygon A to match polygon A perfectly, that is: their dissimilarity should be zero. For this reason we do not scale the polyline or the polygon prior to the matching process. Thus, our dissimilarity measure is not scale-invariant. In our part-based retrieval application (see section 3) we achieve robustness to scaling by normalizing all shapes in the collection to the same diameter of their circumscribed circle.

The turning function is sensitive to unevenly spread noise, since that distorts the parameterization of the curves. Evenly spread noise is less a problem.

2.2 Dissimilarity measure

The new dissimilarity measure we introduce in this paper is based on the turning function, but it is redesigned to measure the dissimilarity between *a set of polylines and a polygon*. Let $\Theta : [0, l] \rightarrow \mathbb{R}$ be the turning function of a polygon P with m vertices, and of perimeter length l . Since P is a closed polygon, the domain of Θ can be easily extended to the entire real line, by $\Theta(s+c) = \Theta(s) + c \cdot 2\pi$, for c any integer. Let $\{P_1, P_2, \dots, P_k\}$ be a set of polylines, and let $\Theta_j : [0, l_j] \rightarrow \mathbb{R}$ denote the turning function of the polyline P_j of length l_j . If P_j is made of n_j segments, Θ_j is piecewise-constant with $n_j - 1$ jumps (see figure 2).

For simplicity of exposition, we denote by $f_j(t, \theta)$ the quadratic dissimilarity $f(P, P_j, t, \theta)$ between the polyline P_j and the polygon P , for a given placement (t, θ) of P_j over P . Thus,

$$f_j(t, \theta) = \int_0^{l_j} (\Theta(s+t) - \Theta_j(s) + \theta)^2 ds.$$

We define a measure for the dissimilarity between an ordered set $\{P_1, P_2, \dots, P_k\}$ of k polylines and a polygon P . We assume the polylines satisfy the condition $\sum_{j=1}^k l_j \leq l$. The dissimilarity measure, which we denote by $d(P_1, \dots, P_k; P)$, is the square root of the sum of quadratic similarities f_j , minimized over all valid placements of P_1, \dots, P_k over P (or in other words, minimized over all valid horizontal and vertical shifts of their turning functions): $d(P_1, \dots, P_k; P) =$

$$\min_{\text{valid placements}} \left(\sum_{j=1}^k f_j(t_j, \theta_j) \right)^{1/2} \\ (t_1, \theta_1) \dots (t_k, \theta_k)$$

It remains to define what the valid placements are. The horizontal shifts t_1, \dots, t_k correspond to shiftings of the starting points of the polylines P_1, \dots, P_k along P . These horizontal shifts cannot be independent of each other, due to the required validity of the match and the ordering condition. The validity of the match is the condition that all k polylines should cover pieces of P that are mutually disjoint except possibly at their endpoints. The ordering condition implies that the starting points of P_1, \dots, P_k are matched with points on the boundary of P in counterclockwise order around P , that is: $t_{j-1} \leq t_j$ for all $1 < j \leq k$, and $t_k \leq t_1 + l$. Furthermore, the validity of the match implies a sharpening of the constraints to $t_{j-1} + l_{j-1} \leq t_j$ for all $1 < j \leq k$, and $t_k + l_k \leq t_1 + l$ (see figure 2). Without loss of generality, we may restrict the possible choices for t_1 to $[0, l)$. Thus, t_2, \dots, t_k must lie in a subinterval of $[0, 2l)$. The vertical shifts $\theta_1, \dots, \theta_k$ correspond to rotations of the polylines P_1, \dots, P_k with respect to the reference orientation, and are independent of each other.

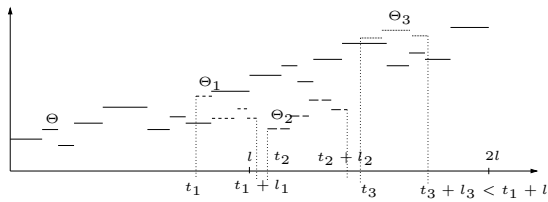


Figure 2: In order to measure the degree of matching $d(P_1, \dots, P_k; P)$ between the polylines P_1, P_2, P_3 and the polygon P , we shift the turning functions Θ_1, Θ_2 , and Θ_3 horizontally and vertically over the turning function Θ .

The dissimilarity measure between the polylines P_1, \dots, P_k and the polygon P is thus given by: $d(P_1, \dots, P_k; P) =$

$$\min_{\substack{t_1 \in [0, l], t_2, \dots, t_k \in [0, 2l]; \\ \forall j \in \{2, \dots, k\}: t_{j-1} + l_{j-1} \leq t_j; \quad t_k + l_k \leq t_1 + l}} \left(\sum_{j=1}^k f_j^*(t_j) \right)^{1/2}, \quad (1)$$

where $f_j^*(t_j) = \min_{\theta \in \mathbb{R}} f_j(t_j, \theta)$ is the quadratic dissimilarity between P_j and P for a given positioning t_j of the starting point of P_j over P , minimized over all rotations of P_j .

The properties of $d(P_1, \dots, P_k; P)$ and the characterization of the optimal solution lead to a straightforward dynamic programming algorithm that runs in $O(km^2n^2)$ time using $O(km^2n^2)$ storage. We also have developed a more efficient algorithm that runs in $O(kmn \log mn)$ time and uses $O(kmn \log mn)$ storage [14].

3. PART-BASED SHAPE RETRIEVAL

In order to demonstrate the effectiveness of the new dissimilarity measure for part-based shape retrieval, we have developed an experimentation system that uses the dissimilarity measure introduced in the previous section. The retrieval problem we are considering is the following: given a large collection of polygonal shapes, and a query consisting of a set of polylines, we want to retrieve those shapes in the collection that best match the query. The query represents a set of disjoint boundary parts of a single shape, and the matching process evaluates how closely these parts resemble pieces of a shape in the collection. Thus, instead of querying with complete shapes, we make the query process more flexible by allowing the user to search for only certain parts. The parts in the query are selected by the user from an automatically computed decomposition of a given contour.

The part-based retrieval works for any decomposition method, but in our implementation of the system, we have used a decomposition based on the medial axis [13].

3.1 Experimental Results

As test collection for the retrieval application we used the MPEG-7 shape silhouette database. We have used the Core Experiment ‘‘CE-Shape-1’’ part B [6], a test set devised by the MPEG-7 group to measure the performance of dissimilarity-based retrieval for shape descriptors. This test set consists of 1400 images: 70 shape classes of 20 images. The outer closed contour of the object in each image was extracted. In this contour, each pixel corresponds to a vertex. In order to decrease the number of vertices, we then used the Douglas-Peucker [3] polygon approximation algorithm. This also alleviates the potential problem of noise.

Each simplified contour was then decomposed into parts. The polygonal decomposition has proven to be robust against the level

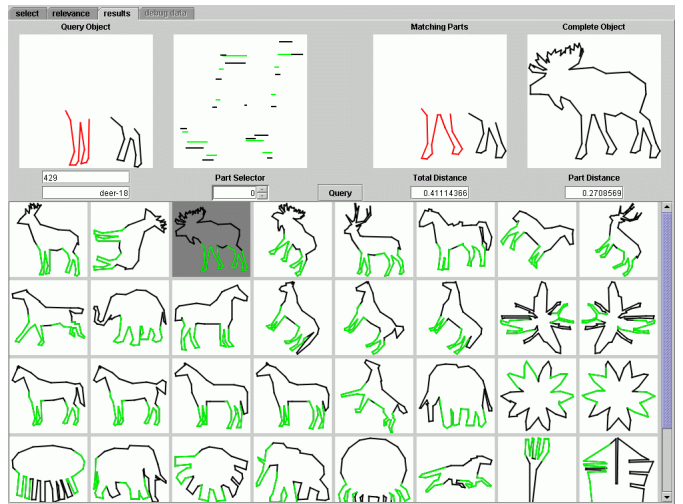


Figure 3: The retrieved results interface of our part-based shape retrieval application.

of polygon approximation. A smaller number of vertices in the approximation has little effect on the simplified medial axis, and thus on the polygon decomposition, see [13].

The matching of a query consisting of k polylines to an arbitrary database contour is based on the dissimilarity measure described in section 2. This dissimilarity measure is not scale invariant, as we noticed in section 2.1. The MPEG-7 shape collection, however, contains shapes at different scales. In order to achieve robustness to scaling, we scaled all shapes in the collection to the same diameter of the circumscribed disk. The reason we opted for a normalization based on the circumscribed disk, instead of the bounding box, for example, is that a class in the collection may contain images of an object at different rotational angles.

In order to formulate a query, the user selects an arbitrary shape in the collection, and then a set of parts from its decomposition. The selected parts can be treated either as separate chains of the query, or as adjacent concatenated parts. Figure 3 depicts the results interface of our experimentation platform. The best 40 matches are shown to the user, who is allowed to select any retrieved shape in order to find out detail information about the matching process. The query parts are depicted in the upper left side of the screen, in red. The selected retrieved shape appears in the upper right side, and its pieces matched by the query appear to its left. Through the dialog box ‘‘Part Selector’’, the user can select a query part. For any retrieved shape, and any query part the system visualizes the turning functions of the query part and the piece of the retrieved shape that is matched to. The quadratic dissimilarity between these polylines appears in the dialog box ‘‘Part Distance’’, while the overall dissimilarity between the query and the selected retrieved shape appears in the dialog box ‘‘Total Distance’’.

The selection of parts comprising the query has a big influence on the results of part-based retrieval. The MPEG-7 collection includes classes ‘‘horse’’, ‘‘dog’’, ‘‘deer’’, ‘‘cattle’’, whose shapes have similar parts, such as limbs. Querying with such parts retrieves shapes from all these classes, see for example figure 3. Though only a few shapes from the same class are ranked in the first 20 retrieved images, the part-based matching results cannot be regarded as poor. If we want to retrieve as many shapes from the same class, the selection of parts should capture more relevant and specific characteristics of the shape.

image	Bull's Eye Performance (%)			True Positives in class size (%)		
	CSS	GTA	PBR	CSS	GTA	PBR
beetle-10	10	35	60	5	20	55
beetle-20	10	30	65	10	10	55
butterfly-4	15	25	55	10	20	55
butterfly-11	20	45	65	20	35	50
bird-9	20	25	50	15	25	40
bird-11	5	20	45	5	15	35
bird-17	20	15	60	20	15	40
carriage-18	70	80	95	45	80	90
crown-13	30	30	50	25	25	40
deer-13	20	40	45	15	40	40
deer-15	15	40	50	5	30	35
dog-11	10	15	50	10	15	45
horse-3	15	25	65	10	25	40
horse-4	25	30	50	20	20	45
horse-17	10	10	70	10	5	60
ray-3	15	15	70	15	5	60
ray-17	15	25	50	15	20	40

Figure 4: Experiment results. A comparison of the Curvature Scale Space (CSS), the global matching based on the turning function (GTA), and our Part-based Retrieval (PBR).

The MPEG-7 Visual Standard is initiated in order to specify standard content-based descriptors that allow to measure dissimilarity in images or video based on visual criteria. Each visual descriptor incorporated in the MPEG-7 Standard was selected from a few competing proposals, based on an evaluation of their performance in a series of tests called Core Experiments. The Core Experiment “CE-Shape-1” was devised to measure the performance of 2D shape descriptors. The performance of several shape descriptors, proposed for standardization within MPEG-7, is reported in [10]. The performance of each shape descriptor was measured using the so-called “bull’s-eye performance”: each image is used as a query, and the number of retrieved images belonging to the same class was counted in the top 40 (twice the class size) matches.

The shape descriptor selected by MPEG-7 to represent a closed contour of a 2D object in an image is based on the Curvature Scale Space (CSS) representation. The reported dissimilarity-based retrieval performance of the CSS described in [9] is 75.44%.

The dissimilarity used by our retrieval application is based on a turning function representation. A whole contour turning function-based shape descriptor is reported to have a dissimilarity-based retrieval performance of 54.14% [5].

For easy classes in “CE-Shape-1”, with a low variance among their shapes, the CSS matching [9] gives good results, with a retrieval rate over 90%, as measured by the “bull’s-eye performance”. For the class labeled “beetle”, however, the different relative lengths of the antennas/legs, and the different shapes of the body pose problems for global retrieval. The average performance rate of CSS matching for this class is only 36%. For the “ray” and “deer” classes, the bad results (an average performance rate of the CSS matching of 26% and 33%, respectively) are caused by the different shape and size of the tails and antlers, respectively, of the contours in these classes. A part-based matching with a proper selection of parts, is significantly more effective in such cases.

We tested the performance of our part-based shape matching. An overall performance percentage for the matching process, like in [10], however, is untractable, since that would require 1400 interac-

tive queries. We therefore present comparison results on a number of individual queries. We compared our approach with the global CSS matching and with a global matching based on the turning function. Figure 4 presents a set of instances when a part-based approach outperforms these global matching methods.

4. CONCLUDING REMARKS

We introduced a new measure for computing the dissimilarity between a set of parts of one shape and another shape. Its effectiveness was demonstrated in a part-based retrieval system. A prerequisite of effectiveness of the part-based matching is the selection of query parts that capture relevant and specific characteristics of the shape. This has to be done interactively by the user. Note that with only slight adaptations we can also solve the problem of matching an ordered set of polylines against an open polyline.

This paper focusses on the dissimilarity measure itself, not on a fully-fledged retrieval system. We have shown, however, that the use of a robust boundary decomposition method allows effective application of our new dissimilarity measure. Experimental results indicate that for those classes with a low average performance of the CSS matching, our approach consistently performs better.

It is difficult to compare our multiple polyline to polygon matching with other part-based matching methods, because they either work on feature vectors, on points, or on a single polyline, and each method requires its own very specific interactive part selection.

The algorithm to compute the dissimilarity measure was implemented in C++, the part-based retrieval interface was written in Java. The running time for a single query on the MPEG-7 test set of 1400 images is typically about one second on a 2 GHz PC.

Acknowledgment

We thank Geert-Jan Giezeman for programming support. This research was supported by the FP6 IST projects 511572-2 PROFI and 506766 AIM@SHAPE, and the Dutch Science Foundation (NWO) project 612.061.006 MINDSHADE.

References

- [1] E. Arkin, L. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell. An efficiently computable metric for comparing polygonal shapes. *PAMI*, 13:209–215, 1991.
- [2] Scott D. Cohen and Leonidas J. Guibas. Partial matching of planar polylines under similarity transformations. In *Proc. SODA*, pages 777–786, 1997.
- [3] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.
- [4] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *PAMI*, 15:850–863, 1993.
- [5] IBM. Technical summary of turning angle shape descriptors proposed by IBM. Technical report, IBM, 1999. TR ISO/IEC JTC 1/SC 29/WG 11/P162.
- [6] S. Jeannin and M. Bober. Description of core experiments for MPEG-7 motion/shape. Technical Report ISO/IEC JTC 1/SC 29/WG 11 MPEG99/N2690, March 1999.
- [7] L. J. Latecki, R. Lakämper, and D. Wolter. Shape similarity and visual parts. In *Proceedings of the International Conference on Discrete Geometry for Computer Imagery (DGCI)*, pages 34–51, 2003.
- [8] H.-C. Liu and M. D. Srinath. Partial shape classification using contour matching in distance transformation. *PAMI*, 12(11):1072–1079, 1990.
- [9] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Workshop on Image DataBases and Multimedia Search*, pages 35–42, 1996.
- [10] J.-R. Ohm and K. Müller. Results of MPEG-7 Core Experiment Shape-1. Technical Report ISO/IEC JTC1/SC29/WG11 MPEG98/M4740, July 1999.
- [11] E. Ozcan and C. K. Mohan. Partial shape matching using genetic algorithms. *Pattern Recognition Letters*, 18(10):987–992, 1997.
- [12] K. Siddiqi, A. Shokoufandeh, S.J. Dickinson, and S.W. Zucker. Shock graphs and shape matching. *IJCV*, 55(1):13–32, 1999.
- [13] Mirela Tanase. *Shape Decomposition and Retrieval*. PhD thesis, Utrecht University, Department of Computer Science, 2005.
- [14] Mirela Tanase, Remco C. Veltkamp, and Herman Haverkort. Multiple polyline to polygon matching. Utrecht University TR UU-CS-2005-017, <http://ftp.cs.uu.nl/pub/RUU/CS/techreps/>. Accepted for ISAAC05.
- [15] Haim Wolfson and Isidore Rigoutsos. Geometric hashing: an overview. *IEEE Computational Science & Engineering*, pages 10–21, October-December 1997.