State-of-the-Art in Shape Matching

Remco C. Veltkamp and Michiel Hagedoorn
Utrecht University, Department of Computing Science
Padualaan 14, 3584 CH Utrecht, The Netherlands
mh@cs.ruu.nl, Remco.Veltkamp@cs.ruu.nl

1 Introduction

Large image databases are used in an extraordinary number of multimedia applications in fields such as entertainment, business, art, engineering, and science. Retrieving images by their content, as opposed to external features, has become an important operation. A fundamental ingredient for content-based image retrieval is the technique used for comparing images. There are two general methods for image comparison: intensity-based (color and texture) and geometry-based (shape). A recent user survey about cognition cognition aspects of image retrieval shows that users are more interested in retrieval by shape than by color and texture [SdLV99]. However, retrieval by shape is still considered one of the most difficult aspects of content-based search. Indeed, systems such as IBM's QBIC, Query By Image Content [QBI], perhaps one of the most advanced image retrieval systems to date, is relatively successful in retrieving by color and texture, but performs poorly when searching on shape. A similar behavior shows the new Alta Vista photo finder [AVP].

Shape matching is a central problem in visual information systems, computer vision, pattern recognition, and robotics. Applications of shape matching include industrial inspection, fingerprint matching, and content-based image retrieval. Figures 1, 2, and 3 illustrate a few typical problems that need to be solved:

1. Figure 1 illustrates an application in agricultural inspection. A typical problem here is to find a matching transformation. Based on shape characteristics, we can find the transformation that matches one piece of fruit with another.

2. Figure 2 shows a point set matching application in fingerprint identification. After extraction of featuring points, two point sets must be matched. The difficulty here is that there is typically no one to one correspondence between the two point sets. The matching technique should be robust against noise and occlusion.

3. Figure 3 shows an application in multimedia retrieval. Given the query shape at the left, the task is to find all pictures that contain similar shapes. The typical problem is that only pieces of the query shape appear in only parts of some of the database pictures.

This paper deals with the matching of geometric shapes, with an emphasis on techniques from computational geometry. We are concerned with geometric patterns such as finite point sets, curves, and regions. For an overview of more general shape analysis, see [Lon98].

Matching deals with transforming a pattern, and measuring the resemblance with another pattern using some dissimilarity measure. Pattern matching and shape matching are commonly used interchangeably. However, more formally, the shape of a pattern is the pattern under all transformations in a transformation group. The matching problem is studied in various forms. Given two patterns and a dissimilarity measure:

- (computation problem) compute the dissimilarity between the two patterns,
- (decision problem) for a given threshold, decide whether the dissimilarity between two patterns is smaller than the threshold,

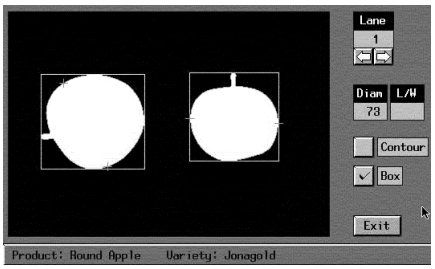

Figure 1: Shape matching in fruit inspection.

- (decision problem) for a given threshold, decide whether there exists a transformation such that the dissimilarity between the transformed pattern and the other pattern is smaller than the threshold,
- (optimization problem) find the transformation that minimizes the dissimilarity between the transformed pattern and the other pattern.

Sometimes the time complexities to solve these problems are rather high, so that it makes sense to devise approximation algorithms that find an approximation:

- Given two patterns, find a transformation that gives a dissimilarity between the two patterns that is within a specified factor from the minimum dissimilarity.

There are several variations on these problems. A pattern can be compared to a single pattern or to many other patterns, in which case an indexing structure is needed to speed up the comparisons. Another variation is to take artefacts such as noise into account, or to perform partial matching, i.e. finding a finding within a larger pattern.

There are various ways to approach the shape matching problem. In this article we focus on methods from computational geometry. Computational geometry is the subarea of algorithms design that deals with the design and analysis of algorithms for geometric problems involving objects like points, lines, polygons, and polyhedra. The standard approach taken in computational geometry is the development of exact, provably correct and efficient solutions to geometric problems. Aspects that play a crucial role in the algorithmic solutions to matching are the representation of patterns, the transformation group, and the dissimilarity measure.

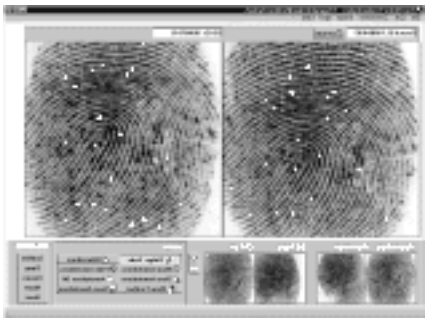

Figure 2: Fingerprint matching.

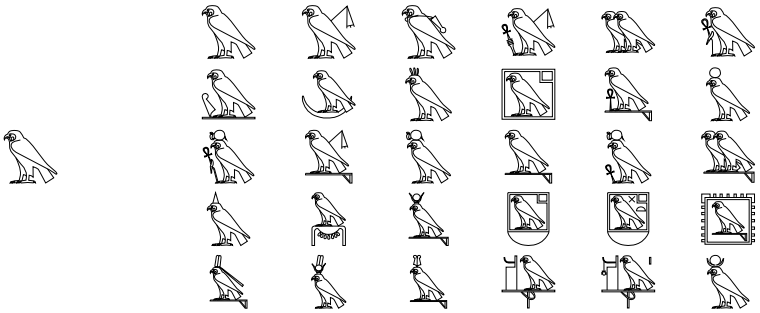

Figure 3: Query hieroglyph (left), and hieroglyphs retrieved from database, from [VV99].

2 Approaches

Matching has been approached in a number of ways, including tree pruning [Ume93], the generalized Hough transform or pose clustering [Bal81] [Sto87], geometric hashing [WR97], the alignment method [HU87], statistics [Sma96], deformable templates [SP95], relaxation labeling [RR80], Fourier descriptors [Lon98], wavelet transform [JFS95], curvature scale space [MAK96], and neural networks [Gol95]. Without being complete, in the following subsections we will describe and group a number of these methods together.

2.1 Global image transforms

There is a number of techniques that transform the image from color information in the spatial domain to color variation information in the frequency domain. Although such approaches do not explicitly encode shape for matching and retrieval, they represent color or intensity transitions in the image, which typically occurs at object boundaries.

A specific class of image transformations are wavelet-based transforms. Wavelets are functions that decompose signals (here two-dimensional color signals) into different frequency components. Each component is then analyzed at a resolution corresponding its scale. Because the original image can be represented as a linear combination of wavelet functions, similar to the Fourier transform, we can process the images by the wavelet coefficients. By truncating the coefficients below a threshold, image data can be sparsely represented, at the cost of loss of detail. A set of such coefficients can be used as a feature vector for image matching.

The wavelet transform can be done with different basis functions. The Haar basis functions, used by Jacobs et al. [JFS95], do not perform well when the query image consist of a small translation of the target image. This problem is less in the approach of Wang et al. using Daubechies basis functions [WWFW97].

For the purpose of shape matching, a drawback of global image transforms is that shape information is not explicitly represented, and that the whole image is encoded, including color and texture information that need not indicate object transitions. As a result, it is not possible to measure how much two different images are similar in terms of shape. Also, due to the global nature, it is not possible to match a query shape with only a part of an image.

2.2 Global Object Methods

Below, we mention a few methods that work on an object as a whole, i.e. a complete object area or contour. An important drawback of all these methods is that complete objects in images must be clearly segmented, which is in itself an ill-posed problem. Typically the result of a segmentation process is a partitioning into regions that need not correspond to whole objects. However, the global object methods work only for whole objects. In general, such methods are not robust against noise and occlusions.

2.2.1 Moments

When a complete object in an image has been identified, it can be described by a set of moments $m_{p,q}$. The (p,q)-moment of an object $O \subseteq \mathbb{R}^2$ is given by

$$m_{p,q} = \int_{(x,y)\in O} x^p y^q \, dx \, dy$$

or, in terms of pixels in a binary $[1,n] \times [1,m]$ image f:

$$\sum_{x=1}^{n} \sum_{y=1}^{m} x^p y^q f(x,y)$$

where the background pixels have value zero, and the object pixels have value one. The infinite sequence of moments, $p,q = 0,1,\ldots$, uniquely determines the shape, and vice verse. Variations are described in [KH90] and [Che93].

Based on such moments, a number of functions, moment invariants, can be defined that are invariant under certain transformations such as translation, scaling, and rotation. Using only a limited number of low order moment invariants, the less critical and noisy high order moments are discarded. A number of such moment invariants can be put into a feature vector, which can be used for matching. Global object features such as area, circularity, eccentricity, compactness, major axis orientation, Euler number, concavity tree, shape numbers, and algebraic moments can all be used for shape description [BB82], [PR92]. A number of such features are for example used by the QBIC system [NBE+93].

2.2.2 Modal matching

Rather than working with the area of an object, the boundary can be used instead. Samples of the boundary can be described with Fourier descriptors, the coefficients of the discrete Fourier transform [vO92].

Another form of shape decomposition is the decomposition into an ordered set of eigenvectors, also called principal components. Again, the noisy high order components can be discarded, using only the most robust components. The idea is to consider n points on the boundary of an object, and to define a matrix D such that element D_{ij} determines how boundary points i and j of the object interact, typically involving the distance between points i and j.

The eigenvectors e_i of D, satisfying $D\,e_i = \lambda e_i$, $i = 1,\ldots,n$, are the *modes* of D, also called eigenshapes. To match two shapes, take the eigenvectors e_i of the query object, and the eigenvectors e'_j of the target object, and compute a mismatch value $m(e_i, e'_j)$. For simplicity, let us assume that the eigenvectors have the same length. For a fixed $i = i_0$, determine the value j_0 of j for which $m(e_{i_0}, e'_j)$ is minimal. If the value of i for which $m(e_i, e'_{j_0})$ is minimal is equal to i_0, then point i of the query and point j of the target match each other. See for example [GT98] and [Scl97] for variations on this basic technique of modal matching.

2.2.3 Curvature scale space

Another approach is the use of a scale space representation of the curvature of the contour of objects. Let the contour C be parameterized by arc-length s: $C(s) = (x(s), y(s))$. The coordinate functions of C are convolved with a Gaussian kernel ϕ_σ of width σ:

$$x_\sigma(s) = \int x(s)\phi_\sigma(t-s)\, dt \qquad \phi_\sigma(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{t^2}{2\sigma^2}}$$

and the same for $y(s)$. With increasing value of σ, the resulting contour gets smoother, see figure 2.2.3, and the number of zero crossings of the curvature along it decreases, until finally the contour is convex and the curvature is positive.

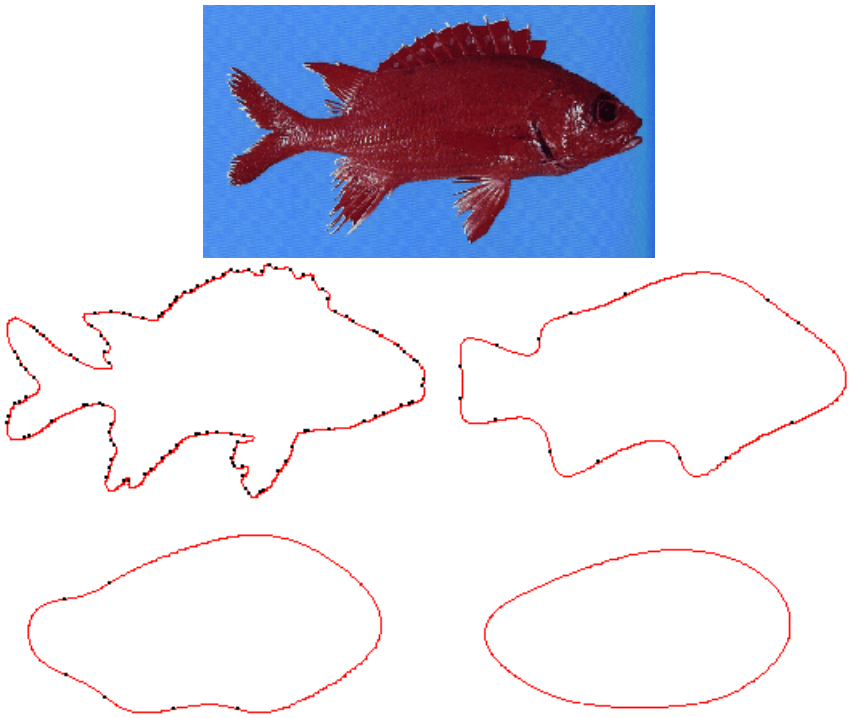

Figure 4: Contour evolution reducing curvature changes, see `http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html`.

For continuously increasing σ, the positions of the curvature zero-crossings continuously move along the contour, until two such positions meet and annihilate. Matching of two objects can be done by matching points of annihilation in the s, σ plane [MAK96].

Another way of reducing curvature changes is based on the turning angle function (see Section 5.1), or tangent space representation [LL99].

2.3 Voting schemes

The voting schemes discussed here generally work on so-called interest points. For the purpose of visual information systems, such points are for example corner points detected in images.

Geometric hashing [LW88, WR97] 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. The method first constructs a single hash table for all target point sets together. Each point is represented as $e_0 + \kappa(e_1 - e_0) + \lambda(e_2 - e_0)$, for some fixed choice of points e_0, e_1, e_2, and the (κ, λ)-plane is quantized into a two-dimensional table, mapping each real coordinate pair (κ, λ) to an integer index pair (k, ℓ).

Let there be N target point sets B_i. For each target point set, the following is done. For each three non-collinear points e_0, e_1, e_2 from the point set, express the other points as $e_0 + \kappa(e_1 - e_0) + \lambda(e_2 - e_0)$, and append the tuple (i, e_0, e_1, e_2) to entry (k, ℓ). If there are $\mathcal{O}(m)$ points in each target point set, the construction of the hash table is of complexity $\mathcal{O}(Nm^4)$.

Now, given a query point set A, choose three noncollinear points e'_0, e'_1, e'_2 from the point set, and express each other point as $e'_0 + \kappa(e'_1 - e'_0) + \lambda(e'_2 - e'_0)$, and tally a vote for each tuple (i, e_0, e_1, e_2) in entry (k, ℓ) of the table. The tuple (i, e_0, e_1, e_2) that receives most votes indicates the target point set T_i containing the query point set. The affine transformation that maps (e'_0, e'_1, e'_2) to the winner (e_0, e_1, e_2) is assumed to be the transformation between the query and the target. The complexity of matching a single query set of n points is $\mathcal{O}(n)$. There are several

variations of this basic method, such as balancing the hashing table, or avoiding taking all possible $\mathcal{O}(n^3)$ 3-tuples.

The generalized Hough transform [Bal81], or pose clustering [Sto87], is also a voting scheme. Here, affine transformations are represented by six coefficients. The quantized transformation space is represented as a six-dimensional table. Now for each triplet of points in the query set, and each triplet of points from the target set, compute the transformation between the two triples, and tally a vote in the corresponding entry of the table. This must be done for all target point sets. The entry with the highest score is assumed to be the transformation between the query and the target. The complexity of matching a single query set is $\mathcal{O}(Nm^3n^3)$.

In the alignment method [HU87, Ull96], for each triplet of points from the query set, and each triplet from the target set, we compute the transformation between them. With each such transformation, all the other points from the target set are transformed. If they match with query points, the transformation receives a vote, and if the number of votes is above a chosen threshold, the transformation is assumed to be the matching transformation between the query and the target. The complexity of matching a single query set is $\mathcal{O}(Nm^4n^3)$.

Variations of these methods also work for geometric features other than points, and for other transformations than affine transformations. A comparison between geometric hashing, pose clustering, and the alignment method is made in [Wol90]. Other voting schemes exist, for example taking a probabilistic approach [Ols97].

2.4 Computational Geometry

Computational geometry is the subarea of algorithms design that deals with geometric problems involving operations on objects like points, lines, polygons, and polyhedra. Over the past twenty years the area has grown into a main-stream world-wide research activity. The success of the field as a research discipline can be explained by the beauty of the problems and their solutions, and by the many applications in which geometric problems and algorithms play a fundamental role. The standard approach taken in computational geometry is the development of exact, provably correct and efficient solutions to geometric problems. See for example the text books [Mul93] [O'R94] [dBvKOS97] [BY98] and the handbook [GO97].

The impact of computational geometry on application domains was minor up to a few years ago. On one hand, the research community has been developing more interest in application problems and real world conditions, and develops more software implementations of the most efficient algorithms available. On the other hand, there is more interest from the application domains in computational geometry techniques, and companies even start to specifically require computational geometry expertise.

Aspects that play an important role in the algorithmic solutions to matching are the representation, decomposition, approximation, and deformation of shapes, the transformation of one shape to another, the measurement of shape similarity, and the organization of shapes into search structures. In the following we give an overview of the state of the art in geometric shape matching from the computational geometry point of view. It should be noted though that the boundary of the field of computational geometry is not sharp, and considering a method a computational geometry method or not is somewhat arbitrary.

First we consider properties of dissimilarity measures, then we list a number of problems in shape matching, together with the best known result to solve them. We are primarily concerned with patterns defined by finite point sets, curves, and regions. Unless otherwise stated, patterns are a subset of $\mathbb{R}^2$, and the underlying distances are Euclidean.

3 Dissimilarity Measures

Many pattern matching and recognition techniques are based on a similarity measure between patterns. A similarity measure is a function defined on pairs of patterns indicating the degree of resemblance of the patterns. It is desirable that such a similarity measure is a metric. Furthermore,

a similarity measure should be invariant for the geometrical transformation group that corresponds to the matching problem. Below, we discuss a number of properties of metrics, such as invariance for transformation groups.

Let S be any set of objects. A *metric* on S is a function $d : S \times S \to \mathbb{R}$ satisfying the following three conditions for all $x, y, z \in S$ [Cop68]:

(i) $d(x, x) = 0$;

(ii) $d(x, y) = 0$ implies $x = y$;

(iii) (triangle inequality) $d(x, y) + d(x, z) \geq d(y, z)$.

If a function satisfies only (i) and (iii), then it is called a *semimetric*. Symmetry follows from (i) and (iii): $d(y, z) \leq d(z, y) + d(z, z) = d(z, y)$, and $d(z, y) \leq d(y, z) + d(y, y) = d(y, z)$, so $d(y, z) = d(z, y)$. An alternative triangle inequality is the following:

(iii') $d(x, y) + d(y + z) \geq d(x, z)$,

but (i) and (iii') do not imply symmetry:

(iv') $d(x, y) = d(y, x)$

So only if d satisfies (iv') in addition to (i) and (iii'), it is a semimetric. Any (semi)metric is nonnegative: $d(x, y) + d(y, x) \geq d(x, x)$, so $d(x, x) \geq 0$.

A set S with a fixed metric d is called a metric space. Given two elements x and y of S, the value $d(x, y)$ is called the *distance* between x and y. By identifying elements of S with zero distance, any semimetric induces a metric on the resulting partition.

A set of bijections G in S is a *transformation group* if $g^{-1}h \in G$ for all $g, h \in G$. A (semi)metric d on a set S is said to be *invariant* for the transformation group G acting on S if $d(g(x), g(y)) = d(x, y)$ for all $g \in G$ and $x, y \in S$.

The *orbit* of G passing through $x \in S$ is the set of images of x under G:

$$G(x) = \{g(x) \mid g \in G\}.$$

The orbits form a partition of S. The collection of all orbits is called the *orbit set*, denoted by S/G.

The following theorem shows that a semimetric invariant under a transformation group results in a natural semimetric on the orbit set. Rucklidge [Ruc96] used this principle to define a shape distance based on the Hausdorff distance.

Theorem 3.1 *Let G be a transformation group for a set S; let d be a semimetric on S invariant for G. Then $\tilde{d} : S/G \times S/G \to \mathbb{R}$ defined by*

$$\tilde{d}(G(x), G(y)) = \inf\{d(g(x), y) \mid g \in G\}$$

is a semimetric.

Let $\mathcal{P}$ be a fixed collection of subsets of $\mathbb{R}^2$. Any element of $\mathcal{P}$ is called a *pattern*. We call the collection $\mathcal{P}$ with a fixed metric d a *metric pattern space*. A collection of patterns $\mathcal{P}$ and a transformation group G determine a family of shapes $\mathcal{P}/G$. For a pattern $A \in \mathcal{P}$, the corresponding *shape* equals the orbit

$$G(A) = \{g(A) \mid g \in G\}.$$

The collection of all these orbits forms a *shape space*. If d is invariant for G, then Theorem 3.1 gives a semimetric $\tilde{d}$ on the shape space $\mathcal{P}/G$.

Shape matching often involves computing the similarity between two patterns, independent of transformation. This is exactly what the shape metric $\tilde{d}$ is good for. Given two patterns A and B,

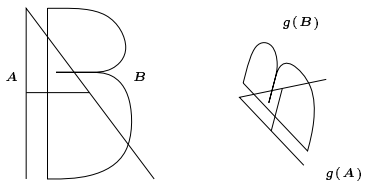

Figure 5: Affine invariance: $d(A, B) = d(g(A), g(B))$.

it determines the greatest lower bound of all $d(g(A), B)$ under transformations $g \in G$, resulting in a transformation-independent distance between the corresponding shapes $G(A)$ and $G(B)$.

A collection of patterns $\mathcal{P}$ uniquely determines a maximal subgroup T of the homeomorphisms under which $\mathcal{P}$ is closed. (Homeomorphims are continuous, bijective functions having a continuous inverse.) The subgroup T consists of all homeomorphism t such that both the image $t(A)$ and the inverse image $t^{-1}(A)$ are members of $\mathcal{P}$ for all patterns $A \in \mathcal{P}$.

The metric pattern space $(X, \mathcal{P}, d)$ is invariant for a transformation $g \in T$ if $d(g(A), g(B))$ equals $d(A, B)$ for all A and B in $\mathcal{P}$. The invariance group G of a metric pattern space consists of all transformations in T for which it is invariant. Affine invariance is often desired in many pattern matching and shape recognition tasks. Figure 5 shows patterns A and B in the Euclidean plane, and image patterns $g(A)$ and $g(B)$ under an affine transformation g. Invariance for affine transformations makes the distance between two patterns independent of the choice of coordinate system.

Finding an affine invariant metric for patterns is not so difficult. Indeed, a metric that is invariant not only for affine transformations, but for general homeomorphisms is the discrete metric:

$$d(A, B) = \begin{cases} 0 & \text{if } A \text{ equals } B \\ 1 & \text{otherwise} \end{cases}$$

However, this metric lacks useful properties. For example, if a pattern A is only slightly distorted to form a pattern A', the discrete distance $d(A, A')$ is already maximal.

Therefore it makes sense to devise metrics with specific properties. A frequently used dissimilarity measure is the Hausdorff distance, which 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. This can be formulated as follows:

$$d(A, B) = \inf\{\epsilon > 0 \mid A \subseteq B^\epsilon \text{ and } B \subseteq A^\epsilon\}$$

where A^ϵ denotes the union of all disks with radius ϵ centered at a point in A. The Hausdorff distance is a metric. The invariance group for the Hausdorff distance consists of isomorphisms (rigid motions and reflections). The Hausdorff distance is robust against small deformations, but it is sensitive to noise: a single outlier, a far away noise point, drastically increases the Hausdorff distance, see Figure 6.

In the next few sections, we give an overview of dissimilarity measures for more restricted patterns: finite point sets, curves, and regions. Then, in Section 7 we will list a number of robustness properties for these measures.

4 Finite point sets

Let A and B be point sets of sizes n and m resp. Matching the point sets means finding a correspondence between points of A and points of B. An optimal matching minimizes some dissimilarity measure between the point sets. The correspondence can be many-to-many, but also one-to-one, both have their applications. Matching has been studied extensively in a graph theory

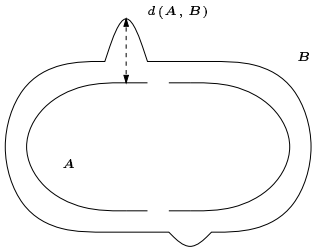

Figure 6: Hausdorff distance.

setting, where the problem is to find a matching in a graph (V, E) with vertices $V = A \cup B$, and given edges E with weights. Exploiting the geometric nature if the vertices are points, and the weights are distances between points, results in more efficient algorithms, see [Vai89] for example.

4.1 Bottleneck matching

Let A and B be two point sets of size n, and $d(a, b)$ a distance between two points. The bottleneck distance is the minimum over all $1-1$ correspondences f between A and B of the maximum distance $d(a, f(a))$. The results on bottleneck distance mentioned in this section are due to [EI96].

If $d(a, b)$ is the Euclidean distance, the bottleneck distance between A and B can be computed in time $\mathcal{O}(n^{1.5} \log n)$. It is computed using a technique called parametric search. This is usually considered an impractical method, although it has been implemented for other problems [SSS97].

An alternative is to compute an approximation d to the bottleneck distance d^*. An approximate matching between A and B with d the furthest matched pair, such that $d^* < d < (1+\epsilon)d^*$, can be computed in time $\mathcal{O}(n^{1.5} \log n)$. This algorithm makes use of an optimal approximate nearest neighbor algorithm [AMN+94].

So far we have considered only the computation problem, computing the distance between two point sets. The decision problem for translations, deciding whether there exists a translation ℓ such that $d(Q + \ell, B) < \epsilon$, can be done in $\mathcal{O}(n^5 \log n)$ time.

Because of the high degree in the complexity, it is interesting to look at approximations with a factor ϵ: $d(Q + \ell, B) < (1+\epsilon)d(Q + \ell^*, T)$. Finding such a translation can be done in $\mathcal{O}(n^{2.5})$ time [Sch92].

The optimization problem considers the computation of the minimum distance under a group of transformations. It finds the optimal transformation f^* such that $d(f(A), B)$ is minimized. For rigid motions (translations plus rotations, sometimes called congruences), this can be found in time $\mathcal{O}(n^6 \log n)$ [AMWW88]. For translations only, it can be computed in time $\mathcal{O}(n^5 \log^2 n)$ [EI96].

An approximation translation ℓ within factor two, $d(A+\ell, B) \leq 2d(A+\ell^*, B)$, can be obtained by translating A such that the lower left corner of the axis parallel bounding box (called reference point) coincides with the one of B. An approximation with factor $1 + \epsilon < 2$ can be obtained in time $\mathcal{O}(C(\epsilon, d)n^{1.5} \log n)$ time, with $C(\epsilon, d)$ a constant depending on ϵ and dimension d: $C(\epsilon, d) = (\frac{1+\epsilon}{\epsilon^2})^d \log(1/\epsilon)$.

Some variations on computing the bottleneck distance between point sets are the following. If A is a set of points, and B a set of segments, computing the bottleneck distance can be done in $\mathcal{O}(n^{1.5+\epsilon})$ time. When the point are in $\mathbb{R}^d$ and the distance is L_∞, it can be computed in time $\mathcal{O}(n^{1.5} \log^{d-1} n)$.

Let A and B be two point sets of size m and n, and k a number not larger than m and n. The problem of finding the smallest bottleneck distance over all one-to-one matchings between k points in A and k points in B can be computed in $\mathcal{O}(m \log m + n^{1.5} \log m)$ time. Typical application of this result is in situations where we search a query pattern A in a larger target pattern B and have to deal with noise points.

4.2 Minimum weight matching

The minimum total distance (weight) is the minimum over all $1-1$ correspondences f between A and B of the sum of the distances $d(a, f(a))$. It can be computed in $\mathcal{O}(n^{2+\epsilon})$ time [AES95]. Here, the constant ϵ stands for a positive constant which can be chosen arbitrarily small with an appropriate choice of other constants of the algorithm. For the L_∞ distance, it can be computed in time $\mathcal{O}(n^2 \log^3 n)$ [Vai89].

4.3 Uniform matching

The 'most uniform' distance is the minimum over all $1-1$ correspondences f between A and B of the difference between the maximum and the minimum $d(a, f(a))$. The most uniform matching is also called balanced or fair matching. The distance can be computed in time $O(n^{10/3} \log n)$ [EK96]. It is based on batched range searching, where the ranges are congruent annuli.

The problem of finding the smallest uniform distance over all one-to-one matchings beteen k points in A and k points in B can be computed with the same time complexity.

4.4 Minimum deviation matching

The minimum deviation distance is the minimum over all $1-1$ correspondences f between A and B of the difference between the maximum and average distance $d(a, f(a))$. This can be computed in time $\mathcal{O}(n^{10/3+\epsilon})$ [EK96].

4.5 Hausdorff distance

In many application, for example stereo matching, not all points from A need to have a corresponding point in B, due to occlusion and noise. Typically, the two point sets are of different size, so that no one-to-one correspondence exists between all points. In that case, a dissimilarity measure that is often used is the Hausdorff distance. The Hausdorff distance was defined in Section 3 for general sets. For finite point sets, it can equivalently be defined as follows.

The *directed* Hausdorff distance $\vec{d}(A, B)$ is defined as the maximum over all points in A of the distances to a point from B. The Hausdorff distance $d(A, B)$ is the maximum of $\vec{d}(A, B)$ and $\vec{d}(A, B)$:

$$d(A, B) = \max\{\vec{d}(A, B), \vec{d}(A, B)\}, \qquad \vec{d}(A, B) = \max_{a \in A} \min_{b \in B} d(a, b)$$

with $d(a, b)$ the underlying (Euclidean, say) distance.

It can be computed using Voronoi diagrams in time $\mathcal{O}((m+n)\log(m+n))$ [ABB95]. The use of Voronoi diagrams for computing the Hausdorff distance is explained in Section 6.3 for matching polygons.

Given two point sets A and B, the translation ℓ^* that minimizes the Hausdorff distance $d(A + \ell, B)$ can be determined in time $\mathcal{O}(mn(\log mn)^2)$ when the underlying metric is L_1 or L_∞ [CK92]. This is done using a search structure called segments tree. For other L_p metrics, $p = 2, 3, \ldots$ it can be computed in time $\mathcal{O}(mn(m+n)\alpha(mn)\log(m+n))$ [HKS93]. ($\alpha(n)$ is the inverse Ackermann function, a very slowly increasing function.) This is done using the upper envelopes of Voronoi surfaces.

Given a real value ϵ, deciding if there is a rigid motion m (translation plus rotation) such that $H(m(A), B) < \epsilon$ can be done in time $\mathcal{O}((m+n)m^2n^2 \log mn)$ [CGH+97]. Computing the optimal rigid motion, minimizing $H(m(A), B)$ can be done in $\mathcal{O}((m+n)^6 \log(mn))$ time [HKK92]. This is done using dynamic Voronoi diagrams.

Given the high complexities of these problems, it makes sense to look at approximations. Computing an approximate optimal Hausdorff distance under translation and rigid motion can be done in time $\mathcal{O}((m+n)\log(m+n))$ [AAR97].

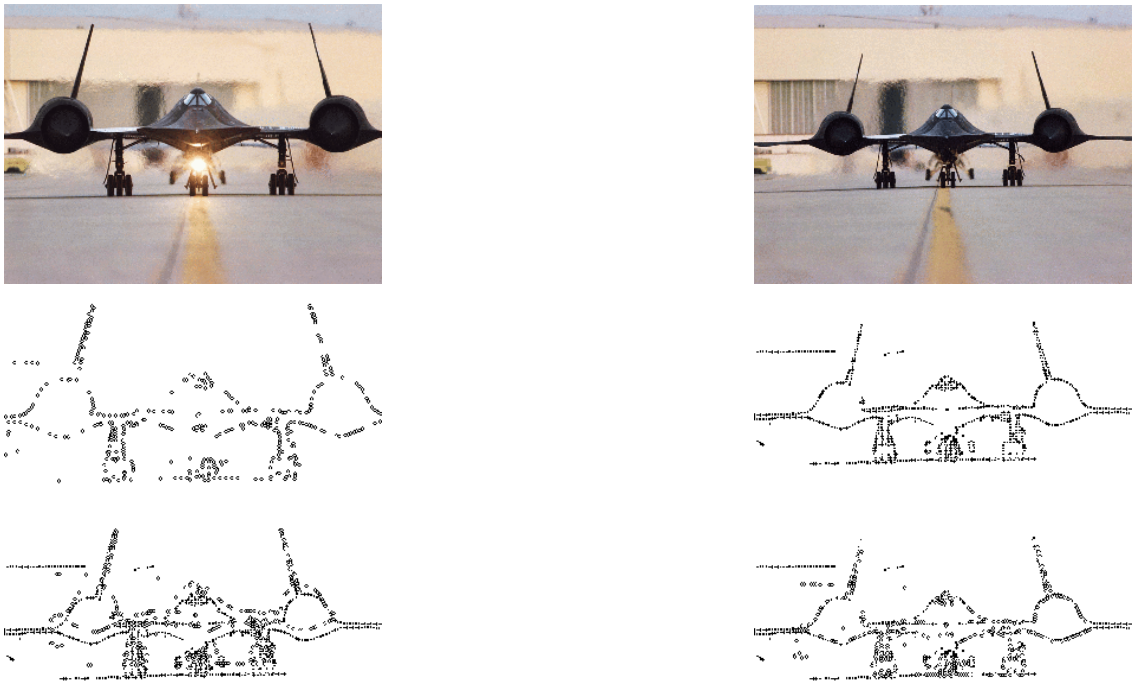

Figure 7: Original images, extracted points, matching with partial Hausdorff distance (lower left), and matching with the affine invariant metric from [HV99b] (lower right).

4.6 Transformation space subdivision

Matching of finite points, from images, under homotheties (translation and scaling) is done by subdividing the transformation space by [HKR93]. Rather than the Hausdorff distance itself, the partial Hausdorff distance is used, which is the maximum of the two directed partial Hausdorff distances $\vec{d}_k(A,B)$ and $\vec{d}_k(B,A)$:

$$d_k(A,B) = \max\{\vec{d}_k(A,B), \vec{d}_k(B,A)\}, \qquad \vec{d}_k(A,B) = \underset{q \in A}{k^{th}} \min_{t \in B} d(a,b)$$

The partial Hausdorff distance is not a metric since it fails the triangle inequality. The running time depends on the depth of subdivision of transformation space.

The subdivision of transformation space is generalized to a general framework by [HV99b]. Here the matching can be done with respect to other transformations as well, for example, similarity (translation, rotation, and scaling), or affine transformation (translation, rotation, scaling, and shear). The method works for many dissimilarity measures, but we used a technique for constructing metrics using functions $\mathbf{f}_A, \mathbf{f}_B : \mathbb{R}^2 \to \mathbb{R}$ defined on patterns A and B, and the affine invariant metric defined by integrating the absolute difference of $\mathbf{f}_A$ en $\mathbf{f}_B$. Figure 7 illustrates matching with this metric, compared to the partial Hausdorff distance.

5 Curves

The most direct way of representing curves is by their position function, defining all the positions of the curve. A parametric curve A is defined in terms of a parameter: $A(t) = (x(t), y(t))$. In general, many parameterizations result in the same shape of the curve, but have different derivative vectors along the curve [Vel92]. A standard parameterization is by arc length along the curve; the arc length is usually denoted by s. Polygonal curves (polylines) are usually represented by

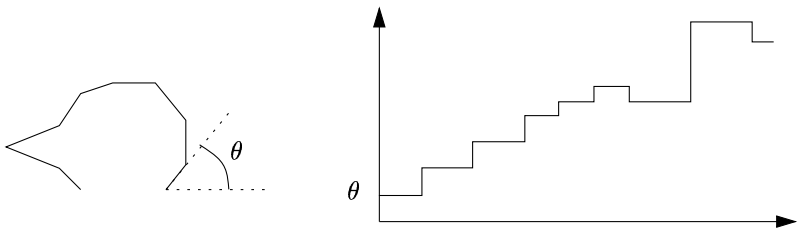

Figure 8: Polygonal curve and turning function.

their sequence of vertices. An implicit definition of the curve, $A : f(x, y) = 0$, is less often used in matching.

Polylines from real world applications often contain many spurious vertices, which can be removed by approximating the polygon. There are many heuristics for approximating polygonal curves, see e.g. [Ros97] for a comparison. Two methods of optimal approximation are the following:

- Given a polyline A and a number k, construct an approximation polyline A_k of k vertices, minimizing the approximation error, or dissimilarity, $d(A, A_k)$.
- Given a polyline and an error bound ϵ, construct an approximation polyline A_ϵ with dissimilarity $d(A, A_\epsilon) < \epsilon$, minimizing the number of vertices.

Both approximations can be computed in $\mathcal{O}(n^2 \log n)$ time for various error measures [II88]. However, these optimal approximations are not suitable for constructing a hierarchy of approximations, in the sense that each segment at one level may be refined at the next level of approximation. Approximating polygons at various levels allows the hierarchical processing of curves [Vel98].

5.1 Turning function

Representations other than the position function are also useful in matching. From the position function, other representations can be derived, such as the tangent, acceleration, tangent angle, cumulative angle, periodic cumulative angle, and the curvature functions [vO92].

The cumulative angle function, or turning function, $\Theta_A(s)$ of a polygon A gives the angle between the counterclockwise tangent and the x-axis as a function of the arc length s. $\Theta_A(s)$ keeps track of the turning that takes place, increasing with left hand turns, and decreasing with right hand turns. Clearly, this function is invariant under translation of the polyline. Rotating a polyline over an angle θ results in a vertical shift of the function with an amount θ.

For polylines, the turning function is a piecewise constant function, increasing or decreasing at the vertices, and constant between two consecutive vertices, see figure 8.

Matching polylines based on the turning functions can be done as follows. For simplicity, first assume that the two curves have the same length. The L_p metric on function spaces, applied to Θ_A and Θ_B, gives a dissimilarity measure on A and B:

$$d_{A,B} = \left(\int |\Theta_A(s) - \Theta_B(s)|^p \, ds \right)^{1/p}$$

Minimizing this dissimilarity under rotation θ, amounts to minimizing $d(A, B) = \int |\Theta_A(s) - \Theta_B(s) + \theta|^p \, ds$. For $p = 2$, the minimum is obtained for $\theta = \int \Theta_B(s) \, ds - \int \Theta_A(s) \, ds$.

In [VV99], for the purpose of retrieving hieroglyphic shapes, the polygonal curves do not have the same length, so that partial matching can be performed. In that case we can move the starting point of the shorter one along the longer one, and consider only the turning function where the arc lengths overlap. This is a variation of the algorithms for matching closed polygons with respect to the turning function, which can be done in $\mathcal{O}(mn \log(mn))$ time [ACH+91], see Section 6.

Partial matching under scaling, in addition to rotation and rotation, is more involved. This can be done in time $\mathcal{O}(m^2 n^2)$, see [CG97]. The dissimilarity balances the length of a match against

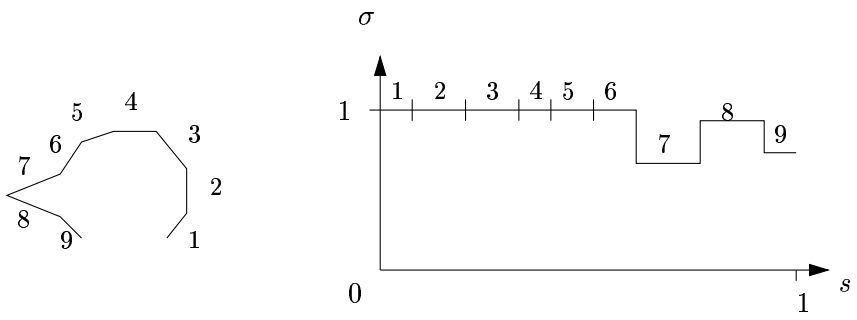

Figure 9: A curve and its signature function.

the squared error. Given two matches with the same squared error, the match involving the longer part of the polylines has a better dissimilarity. The dissimilarity measure is a function of the scale, rotation, and the shift of one polyline along the other. An analytic formula of the dissimilarity in terms of scale and shift yields a search problem in scale-shift plane. This space is divided into regions. A minimum of the dissimilarity is found by a line sweep over the plane.

5.2 Signature function

A less discriminative function is the so-called signature function. At every point along the curve, the signature function σ value is the arc length of the curve to the left or on the tangent line at that point, see Figure 9. It is invariant under similarity: combinations of translation, rotation, and scaling. For convex curves, the signature function is one everywhere, because at every point, the whole curve lies to the left of the tangent. For a single polyline curve, the signature function can be computed in time $\mathcal{O}(n^2)$ [O'R85].

For polylines, dissimilarity measures can be used that are based on 'time warps' of sequences of elements (vertices or segments), pairing elements of A to elements of B. The pairing need not be one-to-one: the pairing of element i of A to element j of B, may be followed by a pairing of i to $j+1$, $i+1$ to j, or $i+1$ to $j+1$. Using dynamic programming, this takes time $\mathcal{O}(nm)$ [Kru83].

5.3 Affine arc-length

Instead of turning functions, affine invariant representations of curves may be used as a basis for shape matching. An example of such a representation is *affine arc-length*. While turning functions are invariant only under similarity transformations the normalized affine arc-length is invariant for all affine transformations. Huttenlocher and Kedem [HK90] use the one-dimensional Hausdorff distance to compare affine arc-length descriptions of curves.

Let $A : [0,1] \to \mathbb{R}$ be a two times continuously differentiable curve, and let A' and A'' denote the first and second order derivates, respectively. The affine arc length is the function $\sigma : \mathbb{R} \to \mathbb{R}$ given by

$$\sigma(t) = \int_0^t |\det(A'(x), A''(x))|^{\frac{1}{3}} \, dx.$$

The *normalised arc-length* is defined as follows:

$$\sigma^*(t) = \frac{\sigma(t)}{\sigma(1)}.$$

Instead of these definitions, Huttenlocher and Kedem use a discretized version of affine arc-length to represent the boundary of a simple polygon. This discretized representation is a finite set of numbers between 0 and 1, one number for each boundary vertex. Two simple polygons are equal if the respective discretized arc-lengths are equal up to translation modulo 1. This problem

can be solved in a perturbation-robust manner by minimising the Hausdorff distance between the two representations (seen as one-dimensional finite point sets). The latter problem can be solved in $\mathcal{O}(mn \log(mn))$ time.

5.4 Reflection metric

Affine-arc length can be used to define affine invariant similarity measures on curves. However, there is no straightforward generalization of it to patterns that consist of more than one connected component. The *reflection metric* ([HV99a]) is an affine-invariant metric that is defined on finite unions of curves in the plane.

The reflection metric is defined as follows. First, unions of curves are converted into real-valued functions on the plane. Then, these functions are compared using integration, resulting in a similarity measure for the corresponding patterns.

The functions are formed as follows, for each finite union of curves A. For each $x \in \mathbb{R}^n$, the *visibility star* V_A^x is defined as the union of open line segments connecting points of A that are visible from x:

$$V_A^x = \bigcup \{ \overline{xa} \mid \ a \in A \text{ and } A \cap \overline{xa} = \varnothing \ \}.$$

The *reflection star* R_A^x is defined by intersecting V_A^x with its reflection in x:

$$R_A^x = \{ \, x + v \in \mathbb{R}^2 \mid \ x - v \in V_A^x \text{ and } x + v \in V_A^x \ \}.$$

The function $\rho_A : \mathbb{R}^2 \to \mathbb{R}$ is the area of the reflection star in each point:

$$\rho_A(x) = \text{area}(R_A^x).$$

Observe that for points x outside the convex hull of A, this area is always zero. The reflection metric between patterns A and B defines a normalised difference of the corresponding functions ρ_A and ρ_B:

$$d(A, B) = \frac{\int_{\mathbb{R}^2} |\rho_A(x) - \rho_B(x)| \, dx}{\int_{\mathbb{R}^2} \max(\rho_A(x), \rho_B(x)) \, dx}.$$

From the definition follows that the reflection metric is invariant under all affine transformations. In contrast with single-curve patterns, this metric is defined also for patterns consisting of multiple curves. In addition, the reflection metric is deformation, blur, crack, and noise robust.

Here, we focus at the computation of the reflection metric for finite unions of line segments in the plane. First, compute partitions of the plane in which the combinatorial structure of the reflection star is constant. Using the latter partition, the reflection distance can be computed in $\mathcal{O}(rI(m+n))$ time for two separate collections of segments with m and n segments, where r is the complexity of the overlay of two partitions, and $I(k)$ denotes the time needed to integrate the absolute value of quotients of polynomials with at most degree k over a triangle. Assuming $I(k)$ is linear in k, the overall complexity amounts to $\mathcal{O}(r(m+n))$. The complexity of the overlay, r, is $\mathcal{O}(m^4 + n^4)$.

The reflection metric can be generalised to finite unions of $(d-1)$-dimensional hyper-surfaces in d dimensions. The generalisation consists of replacing the two-dimensional area by the d-dimensional volume.

5.5 Hausdorff distance

The Hausdorff distance is not only defined for finite point sets, but for any two compact sets. Special cases are sets of polylines. The results for polylines are the same as for polygons, see Section 6.3.

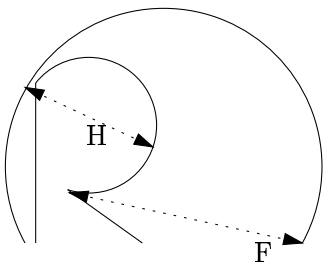

Figure 10: Hausdorff (H) and Fréchet (F) distance between two curves.

5.6 Fréchet distance

The Hausdorff distance is often not appropriate to measure the dissimilarity between curves. For all points on A, the distance to the closest point on B may be small, but if we walk forward along curves A and B simultaneously, and measure the distance between corresponding points, the maximum of these distances may be larger, see Figure 10. This is what is called the Fréchet distance. More formerly, let A and B be two parameterized curves $A(\alpha(t))$ and $B(\beta(t))$, and let their parameterizations α and β be continuous functions of the same parameter $t \in [0,1]$, such that $\alpha(0) = \beta(0) = 0$, and $\alpha(1) = \beta(1) = 1$. The Fréchet distance is the minimum over all monotone increasing parameterizations $\alpha(t)$ and $\beta(t)$ of the maximal distance $d(A(\alpha(t)), B(\beta(t)))$, $t \in [0,1]$.

[AG95] considers the computation of the Fréchet distance for the special case of polylines. Deciding whether the Fréchet distance is smaller than a given constant, can be done in time $\mathcal{O}(mn)$. Based on this result, and the 'parametric search' technique, it is derived that the computation of the Fréchet distance can be done in time $\mathcal{O}(mn \log(mn))$. Although the algorithm has low asymptotic complexity, it is not really practical. The parametric search technique used here makes use of a sorting network with very high constants in the running time. A simpler sorting algorithm leads to an asymptotic running time of $\mathcal{O}(mn(\log mn)^3)$. Still, the parametric search is not easy to implement. A simpler algorithm, which runs in time $\mathcal{O}(mn(m+n)\log(mn))$ is given in [God91].

A variation of the Fréchet distance is obtained by dropping the monoticity condition of the parameterization. The resulting Fréchet distance $d(A,B)$ is a semimetric: zero distance need not mean that the objects are the same, see Section 3. For this the decision problem, deciding whether $d(A,B) < \epsilon$ for a given ϵ, can be decided in time $\mathcal{O}(mn)$. The actual distance can be computed in time $\mathcal{O}(mn \log(mn))$.

Another variation is to consider partial matching: finding the part of one curve to which the other has the smallest Fréchet distance. The corresponding decision problem can be solved in time $\mathcal{O}(mn \log(mn))$, the computation problem in time $\mathcal{O}(mn(\log(mn))^2)$.

5.7 Size function

Relatively new are so-called size functions [VU96]. Size functions can be defined for arbitrary planar graphs and a 'measuring function' D. An example of such a measuring function is the distance from each pattern point to the center of mass. The size function $s_D(x,y)$ is then defined as the number of connected components of the set of points with $D \leq y$ that have at least one point with $D \leq x$. Size functions do not uniquely represent a shape, but classes of shapes, depending on the measuring function.

5.8 Pixel chains

Given two sets of pixel chains, the root mean square of the distances from one set of pixels to the other, can be computed with the relatively efficient hierarchical chamfer matching algorithm, which works on the basis of the distance transform and the chamfer distance [Bor88].

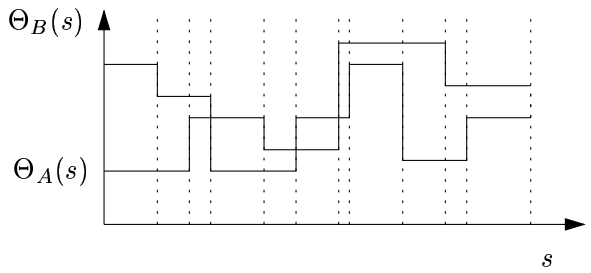

Figure 11: Rectangles enclosed by $\Theta_A(s)$, $\Theta_B(s)$, and dotted lines are used for evaluation of dissimilarity.

6 Regions

As mentioned in Section 2.2.1, normalization of regions, filled contours, is often done using algebraic moments. For the special case of polygons, this can be done in time linear in the number of vertices [Ste96].

A representation that has proven to be relevant in human vision is the medial axis, producing a skeleton and a width value at each point on the skeleton (the so-called quench function). For polygonal contours, the medial axis and the quench function can be computed in time linear in the number of vertices [CSW95]. For pixel chain contours, this can be computed using the distance transform [Bor86].

The dissimilarity of contours can be based on sample points along the contour curve, the whole contour curve, or the enclosed area. For example, Fourier descriptors are based on samples of the contour. A number of methods based on the contour curve and the area are mentioned below.

6.1 Turning function

As already mentioned in Section 5.1, the turning function is also applicable for matching regions, and was used by [ACH+91] for matching polygons under translation, rotation, and scaling. For the special case of polygons, matching based on turning functions can be done as follows. First rescale both polygons so that the perimeter has length one. The L_p metric on function spaces, applied to Θ_A and Θ_B, gives a dissimilarity measure on A and B:

$$d_{A,B} = \left(\int |\Theta_A(s) - \Theta_B(s)|^p \, ds \right)^{1/p}$$

If the starting point of the arc length parameter of $\Theta_A(s)$ is shifted by an amount t, the new function is $\Theta_A(s+t)$. If the polygon is rotated by an angle θ, the new function is $\Theta_A(s)+\theta$. Making the dissimilarity invariant for the starting point of the arc length parameter, and minimizing under rotation θ, amounts to minimizing

$$d_{A,B}(t,\theta) = \left(\int |\Theta_A(s+t) - \Theta_B(s) + \theta|^p \, ds \right)^{1/p}$$

for t and θ.

For any fixed t and $p = 2$, $d_{A,B}(t,\theta)$ is minimal for $\theta = \int \Theta_B(s)\, ds - \int \Theta_A(s)\, ds - 2\pi t$. For polygons, the turning functions are piecewise constant step functions. Therefore $d_{A,B}(t,\theta)$ can be evaluated as the sum of $\mathcal{O}(m+n)$ terms corresponding to the areas between the dotted lines, see figure 11. The minimum $d_{A,B}(t,\theta)$ is obtained when two steps of the step functions coincide, of which are $\mathcal{O}(mn)$ possible solutions. This leads to a straightforward $\mathcal{O}(mn(m+n))$ algorithm. This can be sped up by incremental evaluation of $d_{A,B}(t,\theta)$ for all the $\mathcal{O}(mn)$ possible solutions, giving an algorithm of time complexity $\mathcal{O}(mn\log(mn))$ [ACH+91].

It should be noted that nonuniform noise in the form of perturbation of vertices unevenly spread along the polygon is problematic for this distance function.

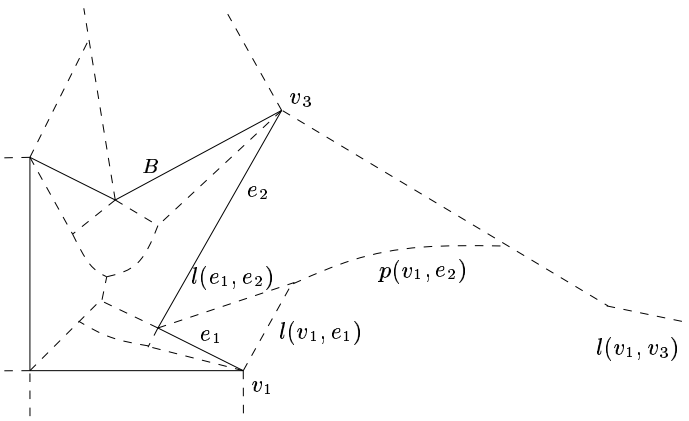

Figure 12: Polygon and its Voronoi diagram.

6.2 Fréchet distance

Parameterized contours are curves where the starting point and ending point are the same. However, the starting and ending point could as well lie somewhere else on the contour, without changing the shape of the contour curve. Deciding whether the Fréchet distance of two contours is smaller than ϵ, irrespective the starting point, can done in time $\mathcal{O}(mn \log(mn))$. The corresponding computation problem, computing the Fréchet distance, can be solved in time $\mathcal{O}(mn(\log(mn))^2)$ [AG95].

For convex contours curves, the Fréchet distance is equal to the Hausdorff distance, which can be computed in time $\mathcal{O}(mn \log(mn))$ [ABGW90].

6.3 Hausdorff distance

Given two polygons A and B, the directed Hausdorff distance from A to B can be computed using the Voronoi diagram of B, which assigns to each vertex and edge of A a region of points that lie closer to that vertex or edge than to any other, see Figure 12. If the edges in the Voronoi diagram separate regions of two edges (e.g. $l(e_1, e_2)$), or two vertices (e.g. $l(v_1, v_3)$), or the regions of an edge and its endpoint vertex (e.g. $l(v_1, e_1)$), then they are line segments. The Voronoi edge is a parabolic segment if it separates regions of a polygon edge and a vertex that not its endpoint (e.g. $p(v_1, e_2)$). The Voronoi diagram of B has $\mathcal{O}(n)$ edges, and it can be computed in time $\mathcal{O}(n \log n)$.

To compute the directed Hausdorff distance from A to B, let us consider the part of B that falls within a single region of the Voronoi diagram of A, for example the thick line segments in Figure 13. Moving along the thick polyline, the distance to B first decreases, than increases, so the maximal distance is obtained at the intersection of the thick segments with the Voronoi diagram. In general, the maximal distance is obtained at a vertex of A or at an intersection point of A with the Voronoi diagram. Note that there can be multiple intersection points on an edge of the Voronoi diagram, and the largest distance is obtained at the intersection with the largest or the smallest coordinates; there are $\mathcal{O}(m+n)$ of these points. At those points of A where the maximal distance can occur, we have to actually compute to distance to B, and take the maximum. This can be done in time $\mathcal{O}((m+n)\log(m+n))$ by a plane sweep algorithm, see [ABB95] for details.

Given two polygons, the minimal Hausdorff distance under translation can be computed in time $\mathcal{O}((mn)^2(\log(m+n))^3)$ using parametric search [AST94], or simpler in time $\mathcal{O}((mn)^3(m+n)\log(m+n))$ [ABB92].

Given the high complexities, it makes sense to implement approximation algorithms to find a transformation that gives a Hausdorff distance that is at most a constant times the minimum distance. For matching under translations, this can be done the following way. Let ℓ_A be the lower left corner of the axis parallel bounding box of A, i.e. it has the smallest x-coordinate of all points in A, and also, independently, the smallest y-coordinate of all points in A. Suppose that the

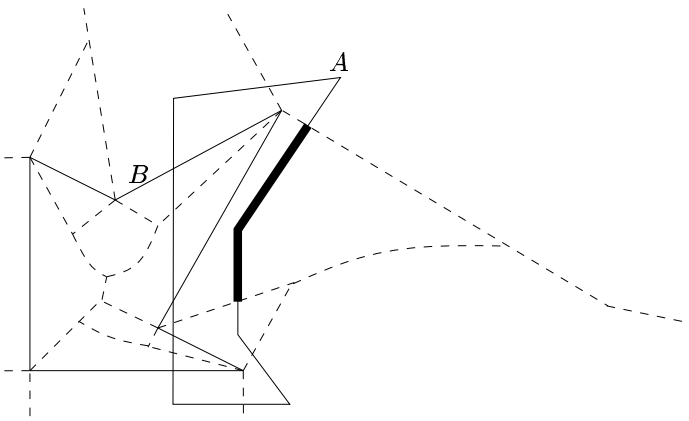

Figure 13: Overlay of polygon A with the Voronoi diagram of B.

optimal translation of A would be f, so that the Hausdorff distance $d_H = d(f(A), B)$ is minimal. Then the distance between ℓ_A and ℓ_B cannot be larger than $d_H\sqrt{2}$. So if g is the translation that maps ℓ_A onto ℓ_B, then the Hausdorff distance $d(g(A), B)$ is at most a factor $(1 + \sqrt{2})$ times the optimal d_H [ABB95]. Determining g can obviously be done in time $\mathcal{O}(m+n)$, but computing the resulting distance still takes $\mathcal{O}(m + n)\log(m + n)$, as above.

The minimal Hausdorff distance under rigid motions (not only translations, but also rotations) can be computed in time $\mathcal{O}((mn)^4(m + n)\log(m + n))$ [ABB92]. So again, an approximation algorithm is interesting. Let k_A be the centroid of the edges of the convex hull of A. Suppose that the optimal rigid motion of A would be f, so that the Hausdorff distance $d_H = d(f(A), B)$ is minimal. There are many rigid motions of A that map k_A onto k_B. If g is the one that gives the smallest Hausdorff distance, then the Hausdorff distance $d(g(A), B)$ is at most a factor $(4\pi + 3)$ times the optimal d_H. For details about how to determine g, see [ABB95]. The time complexity is $\mathcal{O}((mn)\log(mn)\log^*(mn))$. (The notation $\mathcal{O}(\log^* n)$ means $\inf\{k | \log\log \overset{(k \text{ times})}{\cdots} \log n \leq 1\}$). In words, it is the number of times that log has be applied to get down from n to below one. For example $\log^* 2^{4294967296}$ is only 6.)

6.4 Area of overlap and symmetric difference

Two dissimilarity measures that are based on the area enclosed by the polygons rather than the boundaries, are the area of overlap and the area of symmetric difference. For two compact sets A and B, the area of overlap is defined as $area(A \cap B)$, the area of symmetric difference is defined as $area((A-B)\cup(B-A))$, see figure 14. These dissimilarity measure is a metric. The invariance group is the class of diffeomorphisms with unit Jacobi-determinant. For translations, the transformation that maximizes the area overlap also minimizes the area of symmetric difference.

Given two polygons, computing the area of overlap can be done by computing the arrangement of two simple polygons, the combinatorial structure of point, edges, and facets resulting from overlaying the two polygons. This can be done in time $\mathcal{O}(n \log^* n + C)$, with C the complexity of the arrangement (number of vertices, edges, and facets). After preprocessing, taking $\mathcal{O}((mn)^2)$ time, the area of overlap can be computed more efficiently, even for any translation of one polygon with respect to the other, in time $\mathcal{O}(\log(m + n))$ [MW96].

If the polygons are convex, computing the smallest area of overlay under translations can be done in time $O((m + n)\log(m + n))$ [dBDvK$^+$96]. It turns out that translating the polygons so that their centroids coincide gives an overlap of at least 9/25 of the optimal solution [dBDvK$^+$96].

Translating convex polygons so that their centroids coincide also gives an approximate solution for the symmetric difference, which is at most 11/3 of the optimal solution under translations [AFRW96]. This also holds for a set of transformations F other than translations, if the following

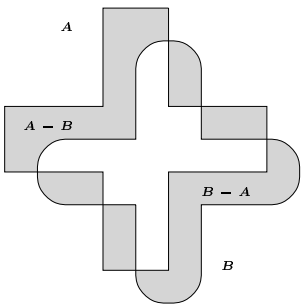

Figure 14: Area of overlap and symmetric difference.

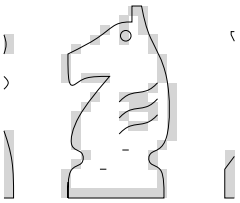

Figure 15: Discretization effects: deformation, blur, cracks, and noise.

holds: the centroid of A, $c(A)$, is equivariant under the transformations, i.e. $c(f(A)) = f(c(A))$ for all f in F, and F is closed under composition with translation.

The computation of the centroids can be done in linear time by triangulating each polygon, determining the centroids and areas of the triangles, and then determining the total centroids as the weighted sum of the triangle centroids. This takes time linear in the number of vertices.

Normalizing the area of overlap and symmetric difference by the area of the union of the two polygons makes these measures invariant under a larger transformation group, namely the group of all diffeomorphisms $f(x)$ with a Jacobi determinant that is constant over all points $x \in \mathbb{R}^2$ [HV99a].

7 Robustness

We have alread seen in Section 3 that the Hausdorff distance is not robust against noise. There are other types of distortions that can also have its effect on the measure of dissimilarity between two patterns. Figure 15 shows the effect discretization can have on a pattern, such as deformation, blurring, as well as the formation of cracks and noise. If we have a robust, invariant metric on patterns, then we can perform shape matching in a robust manner by using the shape metric.

Below, we formalize four types of robustness. We introduce four axioms expressing robustness for what we call 'deformation', 'blur', 'cracks' and 'noise'. Deformation robustness says that each point in a pattern may be moved a little bit without seriously affecting the value of the metric. Blur robustness says that new points may be added close to the original pattern. Crack robustness says that components of patterns may be broken up as long as the cracks are relatively thin. Noise robustness says that new small parts may be added to a pattern.

Let $\mathcal{P}$ be a collection of patterns in $\mathbb{R}^2$, and let T be the maximal group of homeomorphisms under which $\mathcal{P}$ is closed. A metric d on $\mathcal{P}$ is called *deformation robust* if it satisfies the following axiom:

Axiom 7.1 *For each $A \in \mathcal{P}$ and $\epsilon > 0$, there is a $\delta > 0$ such that $\|x - t(x)\| < \delta$ for all $x \in bd(A)$ implies $d(A, t(A)) < \epsilon$ for all $t \in T$.*

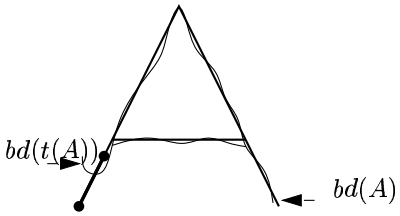

Figure 16: Deformation robust.

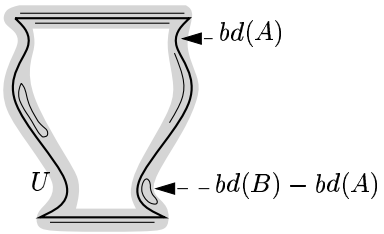

Figure 17: Blur robust.

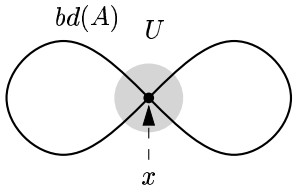

Figure 18: Crack robust.

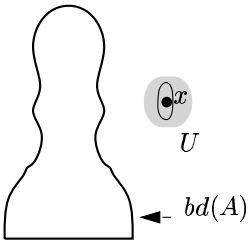

Figure 19: Noise robust.

Deformation robustness is equivalent to saying that for each pattern $A \in \mathcal{P}$, the map $t \mapsto t(A)$ with domain T and range $\mathcal{P}$ is continuous. Figure 16 shows the image of A under a transformation with a small δ in the sense of Axiom 7.1.

In the following, the boundary of a pattern is denoted with $bd(A)$. We call a metric pattern space *blur robust* if the following holds:

Axiom 7.2 *For each $A \in \mathcal{P}$ and $\epsilon > 0$, an open neighborhood U of $bd(A)$ exists, such that $d(A, B) < \epsilon$ for all $B \in \mathcal{P}$ satisfying $B - U = A - U$ and $bd(A) \subseteq bd(B)$.*

The axiom says that additions close to the boundary of A do not cause discontinuities. Figure 17 shows a neighborhood U of A in which parts of B occur that are not in A.

We say that a metric pattern space is *crack robust* if the next axiom holds:

Axiom 7.3 *For each $A \in \mathcal{P}$, each "crack" $x \in bd(A)$, and $\epsilon > 0$, an open neighborhood U of x exists such that $A - U = B - U$ implies $d(A, B) < \epsilon$ for all $B \in \mathcal{P}$.*

The axiom says that applying changes to A within a small enough neighborhood of a boundary point of A results in a pattern B close to A in pattern space. Whether the connectedness is preserved does not matter.

If the following axiom is satisfied, we call a metric pattern space *noise robust*:

Axiom 7.4 *For each $A \in \mathcal{P}$, $x \in \mathbb{R}^2 - bd(A)$, and $\epsilon > 0$, an open neighborhood U of x exists such that $B - U = A - U$ implies $d(A, B) < \epsilon$ for all $B \in \mathcal{P}$.*

This axiom says that changes in patterns do not cause discontinuities in pattern distance, provided the changes happen within small regions. By means of the triangle inequality, we obtain an equivalent axiom when neighborhoods of finite point sets instead of singletons are considered.

Figure 19 shows a pattern A and a point x. Addition of noise $B - A$ within a neighborhood U of x results in a new pattern B. Axiom 7.4 says that the distance between A and B can be made smaller by making U smaller.

All these robustness axioms can also be formulated for patterns in higher dimensions. For a more detailed description, see [HV99c].

For the dissimilarity measures treated in the previous sections, table 1 lists the invariance group, and which robustness axioms are satisfied. For a more detailed treatment, see [HV99a].

distance	pattern	inv. group	deform	blur	crack	noise
bottleneck	finite point sets	iso	yes	n.a.	n.a.	n.a.
minimum weight	finite point sets	iso	yes	n.a.	n.a.	n.a.
most uniform	finite point sets	iso	yes	n.a.	n.a.	n.a.
minimum deviation	finite point sets	iso	yes	n.a.	n.a.	n.a.
Fréchet	curves	iso	yes	n.a.	n.a.	n.a.
turning func + L_p	curves	sim	yes	n.a.	n.a.	n.a.
signature func. + warp	curves	sim	yes	n.a.	n.a.	n.a.
norm. aff. arc length + Hausd.	curves	aff	no	n.a.	n.a.	n.a.
reflection	sets of curves	aff	yes	yes	yes	yes
norm. area symm. diff.	regions	diff-cj	yes	yes	yes	yes
norm. area overlap	regions	diff-cj	yes	yes	yes	yes
Hausdorff	non-empty compact sets	iso	yes	yes	yes	no
discrete	point sets	hom	no	no	no	no

Table 1: Patterns, metrics, invariance group, and robustness. 'Iso' means the group of isomorphisms, 'sim' means similarities, 'diff-cj' means diffeomorphisms with constant Jacobian determinant. 'N.a.' formally means that the axiom is satisfied, but that this is meaningless for that pattern and distance (not applicable).

The distance measure that is most suitable for any particular application totally depends on the application at hand.

8 Software

Most of these results are so recent that almost no implementations are available. Code for matching point sets under the Hausdorff distance is made available via `http://www3.cs.cornell.edu/dph/docs/`. Code for polygon similarity testing using turning angles is available via the Stony Brook Algorithm Repository, see `http://www.cs.sunysb.edu/~algorith/files/shape-similarity.shtml`. Software for size functions and matching with size functions is available via `http://www.dm.unibo.it/~ferri/vismath/sizefcts/sizehom2.htm`. No matching software is available via Netlib (`http://netlib.bell-labs.com/netlib/index.html`), and no matching software at all is mentioned in the overview of computational geometry software of [Ame97].

For implementing geometric algorithms, CGAL, the Computational Geometry Algorithms Library is available via `http://www.cs.uu.nl/CGAL/`. The library provides geometric primitives such as points, lines, and triangles, basic operations such as distance and intersection calculations, as well as higher level data structures and algorithms such as triangulation, convex hull, planar map, etc.

References

[AAR97] O. Aichholzer, H. Alt, and G. Rote. Matching shapes with a reference point. In *International Journal of Computational Geometry and Applications*, volume 7, pages 349–363, August 1997.

[ABB92] Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. In *Proceedings of the 7th Annual ACM Symposium on Computational Geometry*, pages 186–193, 1992.

[ABB95] Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, pages 251–265, 1995.

[ABGW90] H. Alt, J. Blömer, M. Godau, and H. Wagener. Approximation of convex polygons. In *Proceedings of the 17th International Colloquium on Automata, Languages, and Programming (ICALP)*, Lecture Notes in Computer Science 443, pages 703–716. Springer, 1990.

[ACH+91] Esther Arkin, Paul Chew, Daniel Huttenlocher, Klara Kedem, and Joseph Mitchel. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–215, 1991.

[AES95] Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *Proceedings of the 11th Annual ACM Symposium on Computational Geometry*, pages 39–50, 1995.

[AFRW96] Helmut Alt, Ulrich Fuchs, Günter Rote, and Gerald Weber. Matching convex shapes with respect to the symmetric difference. In *Algorithms ESA '96, Proceedings of the 4th Annual European Symposium on Algorithms, Barcelona, Spain, September '96*, pages 320–333. LNCS 1136, Springer, 1996.

[AG95] Helmut Alt and Michael Godeau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, pages 75–91, 1995.

[Ame97] Nina Amenta. *Computational Geometry Software*, chapter 52, pages 951–960. In Goodman and O'Rourke [GO97], 1997.

[AMN+94] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.

[AMWW88] Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237–256, 1988.

[AST94] Pankaj K. Agarwal, Micha Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. *Journal of Algorithms*, 17:292–318, 1994.

[AVP] Alta Vista Photo Finder, `http://image.altavista.com/cgi-bin/avncgi`.

[Bal81] D. H. Ballard. Generalized Hough transform to detect arbitrary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):111–122, 1981.

[BB82] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice Hall, 1982.

[Bor86] G. Borgefors. Distance transforms in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.

[Bor88] Gunilla Borgefos. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 10(6):849–865, November 1988.

[BY98] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry.* Cambridge University Press, 1998.

[CG97] Scott D. Cohen and Leonidas J. Guibas. Partial matching of planar polylines under similarity transformations. In *Proceedings of the 8th Annual Symposium on Discrete Algorithms*, pages 777–786, 1997.

[CGH+97] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. *Computational Geometry, Theory and Applications*, 7:113–124, 1997.

[Che93] C. C. Chen. Improved moment invariants for shape discrimination. *Pattern Recognition*, 26(5):683–686, 1993.

[CK92] Paul Chew and Klara Kedem. Improvements on approximate pattern matching. In *3rd Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science 621, pages 318–325. Springer, 1992.

[Cop68] E. T. Copson. *Metric spaces.* Cambridge University Press, 1968.

[CSW95] F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. In *Proceedings of the 6th Annual International Symposium on Algorithms Computation (ISAAC 95)*, Lecture Notes in Computer Science 1004, pages 382–391. Springer, 1995.

[dBDvK+96] Mark de Berg, Olivier Devillers, Marc van Kreveld, Otfried Schwarzkopf, and Monique Teillaud. Computing the maximum overlap of two convex polygons under translation. In *Proc. 7th Annu. Internat. Sympos. Algorithms Comput.*, 1996.

[dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 1997.

[EI96] Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. *Proceedings of the 12th Symposium on Computational Geometry*, pages 301–310, 1996.

[EK96] Alon Efrat and Matthew J. Katz. Computing fair and bottleneck matchings in geometric graphs. In *Proceedings of the 7th International Symposium on Algorithms and Computation*, pages 115–125, 1996.

[GO97] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC Press, 1997.

[God91] M. Godau. A natural metric for curves - computing the distance for polygonal chains and approximation algorithms. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*, Lecture Notes in Computer Science 480, pages 127–136. Springer, 1991.

[Gol95] Steven Gold. *Matching and Learning Structural and Spatial Representations with Neural Networks.* PhD thesis, Yale University, 1995.

[GT98] Bilge Günsel and A. Murat Tekalp. Shape similarity matching for query-by-example. *Pattern Recognition*, 31(7):931–944, 1998.

[HK90] D. P. Huttenlocher and K. Kedem. Computing the minimum Hausdorff distance for point sets under translation. In *Proc. 6th Annual ACM Symp. Computational Geometry*, pages 340–349, 1990.

[HKK92] D. P. Huttenlocher, K. Kedem, and J. M. Kleinberg. On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane. In *Proceedings of the 8th Annuual ACM Symposium on Computational Geometry*, pages 110–120, 1992.

[HKR93] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machinen Intelligence*, 15:850–863, 1993.

[HKS93] Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9:267–291, 1993.

[HU87] D. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proceedings of the International Conference on Computer Vision, London*, pages 102–111, 1987.

[HV99a] M. Hagedoorn and R. C. Veltkamp. Metric pattern spaces. Technical Report UU-CS-1999-03, Utrecht University, 1999.

[HV99b] Michiel Hagedoorn and Remco C. Veltkamp. Reliable and efficient pattern matching using an affine invariant metric. *International Journal of Computer Vision*, 31(2/3):203–225, 1999.

[HV99c] Michiel Hagedoorn and Remco C. Veltkamp. A robust affine invariant metric on boundary patterns. *International Journal of Pattern Recognition and Pattern Analysis*, 13(7), November 1999. Special Issue on Invariants for Pattern Recognition and Classification.

[II88] Hiroshi Imai and Masa Iri. *Polygonal Approximation of a Curve – Formulations and Algorithms*, pages 71–86. North-Holland, 1988.

[JFS95] C. Jacobs, A. Finkelstein, and D. Salesin. Fast multiresolution image querying. In *Computer Graphics Proceedings SIGGRAPH*, pages 277–286, 1995.

[KH90] A. Khotanzad and Y. H. Hong. Invariant image recongnition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):489–497, 1990.

[Kru83] J. B. Kruskal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM Review*, 25:201–237, 1983.

[LL99] Longing Jan Latecki and Rolf Lakämper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73(3):441–454, 1999.

[Lon98] Sven Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[LW88] Y. Lamdan and H. J. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. In *2nd Inter. Conf. on Comput. Vision*, pages 238–249, 1988.

[MAK96] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Image Databases and Multi-Media Search, proceedings of the First International Workshop IDB-MMS'96, Amsterdam, The Netherlands*, pages 35–42, 1996.

[Mul93] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, 1993.

[MW96] David M. Mount and Angela Y. Wu. On the area of overlap of translated polygons. *Computer Vision and Image Understanding*, 64:53–61, July 1996.

[NBE+93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: Querying imag by content using color, texture, and shape. In *Electronic Imaging: Storage and Retrieval for Image and Video Databases*, Proceedings SPIE, volume 1908, pages 173–187, 1993.

[Ols97] Clark F. Olson. Efficient pose clustering using a randomized algorithm. *International Journal of Computer Vision*, 23(2):131–147, 1997.

[O'R85] Joseph O'Rourke. *Curve Similarity via Signatures*, pages 295–317. North-Holland, 1985.

[O'R94] Joseph O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1994.

[PR92] Richard J. Prokop and Anthony P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphics Models and Image Processing*, 54(5):438–460, 1992.

[PS85] Franca P. Preparata and Michael Ian Shamos. *Computational Geometry, an Introduction.* Springer-Verlag, 1985.

[QBI] QBIC project, `http://wwwqbic.almaden.ibm.com/`.

[Ros97] Paul L. Rosin. Techniques for assessing polygonal approximations of curves. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 19(5):659–666, 1997.

[RR80] S. Ranade and A. Rosenfeld. Point pattern matching by relaxation. *Pattern Recognition*, 12:269–275, 1980.

[Ruc96] W. Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance.* Lecture Notes in Computer Science. Springer, 1996.

[Sch92] Stefan Schirra. Approximate decision algorithms for approximate congruence. *Information Processing Letters*, 43:29–34, 1992.

[Scl97] Stan Sclaroff. Deformable prototypes for encoding shape categories in image databases. *Pattern Recognition*, 30(4):627–641, 1997.

[SdLV99] Lambert Schomaker, Edward de Leau, and Louis Vuurpijl. Using pen-based outlines for object-based annotation and image-based queries. In D. P. Huijsmans and A. W. M. Smeulders, editors, *Visual Information and Information Systems – Proceedings of the Third International Conference VISUAL'99, Amsterdam, The Netherlands, June 1999*, LNCS 1614, pages 585–592. Springer, 1999.

[Sma96] Christopher G. Small. *The Statistical Theory of Shapes.* Springer Series in Statistics. Springer, 1996.

[SP95] Stan Sclaroff and Alex P. Pentland. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6), June 1995.

[SSS97] J. Schwerdt, M. Smid, and S. Schirra. Computing the minimum diameter for moving points: An exact implementation using parametric search. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, pages 466–468, 1997.

[Ste96] Carsten Steger. On the calculation of arbitrary moments of polygons. Technical report, Dept. Computer Science, University of München, October 1996.

[Sto87] G. Stockman. Object recognition and localization via pose clustering. *Computer Vision, Graphics, and Image Processing*, 40(3):361–387, 1987.

[Ull96] Shimon Ullman. *High-Level Vision.* MIT Press, 1996.

[Ume93] S. Umeyama. Parameterized point pattern matching and its application to recognition of object families. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):136–144, 1993.

[Vai89] Pravin M. Vaidya. Geometry helps in matching. *SIAM Journal of Computing*, 18(6):1201–1224, 1989.

[Vel92] Remco C. Veltkamp. Survey of continuities of curves and surfaces. *Computer Graphics Forum*, 11(2):93–112, 1992.

[Vel98] Remco C. Veltkamp. Hierarchical approximation and localization. *The Visual Computer*, 14(10):471–487, 1998.

[vO92] Peter J. van Otterloo. *A Contour-Oriented Approach to Shape Analysis.* Hemel Hampstead, Prentice Hall, 1992.

[VU96] A. Verri and C. Uras. Metric-topological approach to shape recognition and representation. *Image Vision Computing*, 14:189–207, 1996.

[VV99] Jules Vleugels and Remco C. Veltkamp. Efficient image retrieval through vantage objects. In D. P. Huijsmans and A. W. M. Smeulders, editors, *Visual Information and Information Systems – Proceedings of the Third International Conference VISUAL'99, Amsterdam, The Netherlands, June 1999*, LNCS 1614, pages 575–584. Springer, 1999.

[Wol90] H. J. Wolfson. Model-based object recognition by geometric hashing. In *Proceedings of the 1st European Conference on Computer Vision*, Lecture Notes in Computer Science 427, pages 526–536. Springer, 1990.

[WR97] Haim Wolfson and Isidore Rigoutsos. Geometric hashing: an overview. *IEEE Computational Science & Engineering*, pages 10–21, October-December 1997.

[WWFW97] J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei. Wavelet-based image indexing techniques with partial sketch retrieval capability. In *Proceedings of the Fourth Forum on Research and Technology Advances in Digital Libraries.* IEEE, 1997.