Discrete Geometric Shapes: Matching, Interpolation, and Approximation
A Survey

Helmut Alt
Institut für Informatik
Freie Universität Berlin
D-14195 Berlin
Germany

Leonidas J. Guibas
Computer Science Department
Stanford University
Stanford, CA 94305
USA

December 3, 1996

Abstract

In this survey we consider geometric techniques which have been used to measure the similarity or distance between shapes, as well as to approximate shapes, or interpolate between shapes. Shape is a modality which plays a key role in many disciplines, ranging from computer vision to molecular biology. We focus on algorithmic techniques based on computational geometry that have been developed for shape matching, simplification, and morphing.

1 Introduction

The matching and analysis of geometric patterns and shapes is of importance in various application areas, in particular in computer vision and pattern recognition, but also in other disciplines concerned with the form of objects such as cartography, molecular biology, and computer animation.

The general situation is that we are given two objects A, B and want to know how much they *resemble* each other. Usually one of the objects may undergo certain transformations like translations, rotations or scalings in order to be *matched* with the other as well as possible. Variants of this problem include partial matching, i.e. when A resembles only some part of B, and a data structures version where, for a given object A, the most similar one in a fixed preprocessed set of objects has to be found, e.g. in character or traffic sign recognition. Another related problem is that of *simplification* of objects. Namely, given an object A find the most simple object A' resembling A within a given tolerance. For example, A could be a smooth curve and A' a polygonal line with as few edges as possible.

We also will discuss *shape interpolation* ("morphing"), a problem that has become very interesting recently, especially in computer animation. The objective is to find for two given shapes A and B a continuous transformation that transforms A into B via natural intermediate shapes.

First it is necessary to formally define the notions of objects, resemblance, matching, and transformations.

Objects are usually finite sets of points ("point patterns") or "shapes" given in two dimensions by polygons. Generalizations to, for example, polyhedral surfaces in three and higher dimensions are possible, but most of the work has concentrated on two or three dimensions.

In order to measure "resemblance" various distance functions have been used, in particular much work has been based on the so-called *Hausdorff distance*.

For two compact subsets A, B of the d-dimensional space $\mathbb{R}^d$, we define the *one-sided* Hausdorff distance from A to B as

$$\tilde{\delta}_H(A, B) = \max_{a \in A} \min_{b \in B} ||a - b|| \, ,$$

where $|| \cdot ||$ is the Euclidean distance in $\mathbb{R}^d$ (if not explicitly stated otherwise). The (bidirectional) Hausdorff distance between A and B then is defined as

$$\delta_H(A, B) = \max\left(\tilde{\delta}_H(A, B), \tilde{\delta}_H(B, A)\right) .$$

The Hausdorff distance simply assigns to each point of one set the distance to its closest point on the other and takes the maximum over all these values. It performs reasonably well in practice but may fail if there is noise in the images. An variant intended to be more robust will be presented in Section 2.2.3.

What kind of geometric *transformations* are allowed to match objects A and B depends on the application. The most simple kind are certainly *translations.* The matching problem usually becomes much more difficult if we allow rotations and translations (these transformations are called *rigid motions*, or *Euclidean transformations*). In most cases *reflections* can be included as well without any further difficulty.

Scaling means the linear transformation that "stretches" an object by a certain factor λ about the origin and is represented by the matrix $\left(\begin{smallmatrix} \lambda & 0 \\ 0 & \lambda \end{smallmatrix}\right)$ in two dimensions. We call combinations of translations and scalings *homotheties* and combinations of Euclidean transformations and scalings *similarities.* The most general kind of transformations we will consider are arbitrary *affine transformations* which can occur e.g. in orthographic 2–dimensional projections of 3–dimensional objects.

Considerable research on these topics has been done in computational geometry in recent years. This chapter will give a survey on these results.

2 Point Pattern Matching

In this section we present a variety of geometric techniques for matching points sets exactly or approximately, under some allowed transformation group. We discuss methods of both theoretical and practical interest.

2.1 Exact Point Pattern Matching

A seemingly very natural question is whether two finite sets $A, B \subset \mathbb{R}^d$ of n points each can be matched *exactly* by say, rigid motions, i.e. whether are A and B *congruent*. Of course unless we assume that the input consists of points on a grid, this problem is numerically very unstable. Nevertheless, studying it assuming a "real RAM" model of computation gives some insight in the nature of matching problems and may help in designing algorithms for more realistic cases.

In two dimensions exact point-pattern matching can easily be reduced to string matching, as is shown by the following algorithm which was invented independently by several authors, for example Atkinson [Atk87].

1. Determine the centroids c_A, c_B (i.e. arithmetic means) of the sets A and B, respectively.
2. Determine the polar coordinates of all points in A using c_A as the origin. Then sort A lexicographically with respect to these polar coordinates (angle first, length second) obtaining a sequence $(\phi_1, r_1), \ldots, (\phi_n, r_n)$. Let u be the sequence $(\psi_1, r_1), \ldots, (\psi_n, r_n)$ where $\psi_i = \phi_i - \phi_{(i+1) \bmod n}$. Compute in the same way the corresponding sequence v of the set B.
3. Determine whether v is a cyclic shift of u, i.e. a substring of uu by some fast string-matching algorithm.

It is easy to see that A and B are congruent exactly if the algorithm gives a positive answer. The running time is $O(n \log n)$ because of the sorting in step 2; all other operations take linear time.

For exact point pattern matching in three dimensions the following algorithm is given by Alt et al. [AMWW88]:

1. Determine the centroid c_A and project all points of A onto the unit sphere around c_A obtaining a set A' of points on the sphere. Label each point $a \in A'$ with the sorted list of distances from c_A of all points that have been mapped onto a.
2. Compute the 3-d convex hull C_A of A'.
3. In addition to the labeling of step 2 attach to each point $a \in A'$ an adjacency list of vertices connected to a by an edge of C_A sorted in clockwise order (seen from outside). This list should contain all distances of a to adjacent points and all angles between neighboring edges.
4. Execute steps 1-3 with set B, as well.

5. The hulls C_A and C_B can be considered as labelled planar graphs. The point sets A and B are congruent exactly if these graphs are isomorphic. This isomorphism can be decided by a variant of the partition algorithm of Hopcroft (see [AHU74], section 4.13).

A detailed analysis shows that the running time of this algorithm is $O(n \log n)$. Using similar techniques it can be shown that the matching problem in arbitrary dimension d can be reduced to n problems in dimension $d - 1$.

Consequently we have that the exact point pattern matching problem can be solved for patterns of n points in time $O(n \log n)$ in 2 dimensions and in time $O(n^{d-2} \log n)$ for arbitrary dimension $d \geq 3$.

An alternative approach yielding the same bound for dimension 3 was developed by Atkinson [Atk87].

Concerning transformations other than rigid motions, in some cases there are obvious optimal algorithms for exact point pattern matching in arbitrary dimensions. For *translations*, for example, it suffices to match those two points with the lexicographically smallest coordinate vectors and then to check, whether the other points match as well. If *scaling* of the pattern B to be matched is allowed, one can first determine the *diameters* d_A, d_B of both sets. Their ratio d_A/d_B gives the correct scaling factor for B. Therefore, there is an easy reduction of *homotheties* to translations and of *similarities* to rigid motions. *Reflections* can easily be incorporated by trying to match the set B as well as the set B' which is B reflected through some arbitrary hyperplane, for example, $x_1 = 0$.

Exact point pattern matching under arbitrary *affine transformations* is considered by Sprinzak and Werman [SW94]. First the sets A and B are brought into "canonical form" by transforming their second moment matrices into unit matrices. Then it is shown that A, B can be matched under affine transformations exactly if their canonical forms can be matched under rotations. Since the canonical forms can be computed in linear time the asymptotic time bounds for matching under linear transformations are the same as the ones for rigid motions described above.

2.2 Approximate Point Pattern Matching

More realistic than exact point pattern matching is approximate point pattern matching. Here, given two finite sets of points A, B, the problem is to find a transformation matching each point $b \in B$ into the ε-neighborhood ($\varepsilon \geq 0$) of some point $a \in A$. On the other hand each point in A should lie in the ε-neighborhood of some transformed point of B. Clearly, there are many variants to this problem. The first distinction we make is whether A and B must have the same number of points and the matching must be a one-one-mapping, or whether several points in one set may be matched to the same point in the other. Obviously in the latter case we consider matching with respect to the *Hausdorff-distance*.

2.2.1 One-to-one matching

Alt et al. [AMWW88] give polynomial time algorithms for many variants of one-one-matching of finite point sets. These variants are obtained by the following characteristics:

- different types of transformations that are allowed
- solving either the decision problem: given ε, is there a matching?
  or the optimization problem: find the minimal ε allowing a matching.
- a fixed one-one-mapping between A and B is either already given or one should be found.
- different metrics, a concept generalized by Arkin et al. [AKM+92] to arbitrary "noise regions" around the points.

We will demonstrate the techniques used with the example of solving the decision problem, for a given ε as a Euclidean tolerance, of matching under arbitrary rigid motions without a predetermined one-one-mapping between the point sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$.

First, it can be shown by an easy geometric argument that, if there exists a valid matching of B to A then there is one where two points b_i, b_j of B are matched exactly to the boundaries of the ε-neighborhoods $U_\varepsilon(a_k)$, $U_\varepsilon(a_l)$ of two points in A. Consider this configuration for all 4-tuples of points a_k, a_l, b_i, b_j. Mapping b_i, b_j onto the boundaries of $U_\varepsilon(a_k)$ and $U_\varepsilon(a_l)$ respectively in general leaves one degree of freedom which is parametrized by the angle $\phi \in [0, 2\pi)$ between the vector $b_i - a_k$ and a horizontal line. Considering any other point $b_m \in B$, $m \neq i, j$ for all possible values of ϕ, that point will trace an algebraic curve C_m (of degree 6, in fact; see Figure 1).

Being an algebraic curve of constant degree, any C_m intersects the boundary of any $U_\varepsilon(a_r)$ at most a constant number of times, in fact, at most 12 times. So there are at most 6 intervals of the parameter ϕ where the image of b_m lies inside $U_\varepsilon(a_r)$. All interval boundaries of this kind are collected. They partition the parameter space $[0, 2\pi)$ into $O(n^2)$ intervals, so that for all ϕ in one interval the same points of B are mapped into the same neighborhoods of points of A. All these relationships are represented as edges in a bipartite graph whose two sides of nodes are A and B. Clearly, the decision problem has a positive solution exactly if there is some ϕ for which the corresponding graph has a perfect matching. This is checked by finding the graph for the first subinterval of $[0, 2\pi)$ and constructing a maximum matching for it. Then, while traversing the subintervals from left to right, the maximum matching is updated until a perfect matching is found or it turns out that none exists.

Observe, that this procedure is carried out $O(n^4)$ times for all 4-tuples a_k, a_l, b_i, b_j. A detailed analysis shows that the total running time of the algorithm is $O(n^8)$. In addition, determining the intersection points of the curves of degree 6 with circles could cause

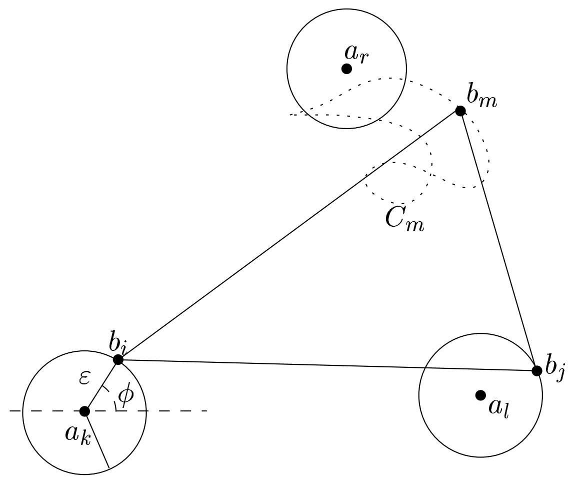


Figure 1: Curve of point b_m when b_i, b_j are moved on the boundaries of $U_\varepsilon(a_k)$, $U_\varepsilon(a_e)$.

nontrivial numerical problems. However, simpler and faster algorithms were found for easier variants of the one-one-matching problem. For the case of translation only, Efrat and Itai [EI96] improve the bounds of [AMWW88] using geometric arguments to speed up the bipartite graph matching involved (for fixed sets, they can compute what they call the optimum *bottleneck matching* in time $O(n^{1.5} \log n)$). Arkin et al. [AKM$^+$92] give numerous efficient algorithms mostly assuming that the ε-neighborhoods or other noise-regions of the points are disjoint. For example, the problem considered above is shown to be solvable in $O(n^4 \log n)$ time under this assumption. Also a generalization from rigid motions to similarity transformations is given in that article.

Heffernan and Schirra [HS92a] take an alternative approach to reduce the complexity of the decision problem in point pattern matching, which they call *approximate decision algorithms*. They only require the algorithm to give a correct answer if the given tolerance ε is not too close to the optimal solution; more precisely, it has to lie outside the interval $[\varepsilon_{opt} - \alpha, \varepsilon_{opt} + \beta]$ for fixed $\alpha, \beta \geq 0$. This way, using network flow algorithms, they can reduce the running time for solving the problem described above to $O(n^{2.5})$. Behrends [Beh90] also considers approximate decision algorithms. Assuming in addition that the ε-neighborhoods are disjoint, he obtains a running time of $O(n^2 \log n)$. The best results in the case that the mapping between A and B is predetermined, are due to Imai, Sumino, and Imai [ISI89] who analyze the lower envelope of multivariate functions in order to find the optimal solution.

2.2.2 Point pattern matching with respect to Hausdorff-distance

Now A and B may have different cardinalities, let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$. The Hausdorff-distance between A and B can be computed straightforwardly in time

a_1 a_3 a_2 a_4

$O(nm)$. It is more efficient to construct the Voronoi-diagrams $VD(A)$ and $VD(B)$ and to locate each point of A in $VD(B)$ and vice versa in order to determine its nearest neighbor in the other set. This way the running time can be reduced to $O((n+m)\log(n+m))$ (see [ABB91]).

Algorithms for optimally *matching* A, B under translations for arbitrary L_p-metrics in 2 and 3 dimensions using Voronoi diagrams are given by Huttenlocher, Kedem, and Sharir [HKS93]. The idea of these algorithms is as follows:

The *Voronoi-surface* of the set A is the graph of the function

$$d(x) = \min_{a \in A} \|x - a\|$$

which assigns to each point x the distance to the nearest point in A. Clearly, $d(x)$ is the *lower envelope* of all $d_a(x) = \|x - a\|$, where $a \in A$. For example, for L_2 and dimension 2 the graph of $d_a(x)$ is an infinite cone in 3-dimensional space whose apex lies in a (see Figure 2).

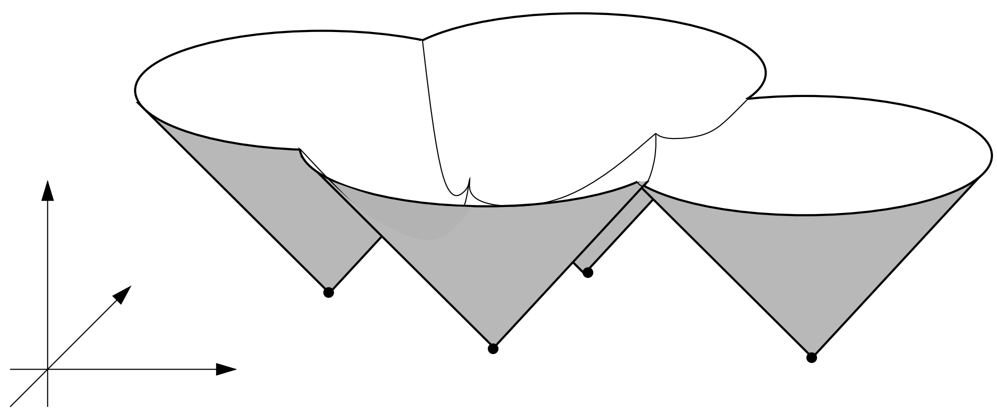


Figure 2: Voronoi surface of A.

The graph of $d(x)$ is piecewise composed of these cones and the projection of the boundaries of these pieces is the Voronoi diagram of A.

If B is translated by some vector t the distance of any $b \in B$ to its nearest neighbor in A is

$$\delta_b(t) = \min_{a \in A} \|a - (b + t)\| = \min_{a \in A} \|(a - b) - t\| = d_{a-b}(t)$$

so the graph of δ_b is the Voronoi surface of A translated by the vector $-b$. The directed Hausdorff distance $\tilde{\delta}_H(B + t, A)$ is the function

$$f(t) = \max_{b \in B} \delta_b(t)$$

and, consequently, the upper envelope of m Voronoi surfaces, namely those of $A - b_1, A - b_2, \ldots, A - b_m$. On the other hand we consider

$$g(t) = \tilde{\delta}_H(A, B + t).$$

Since $g(t) = \tilde{\delta}_H(A + (-t), B)$ we can define $g(t)$ by upper envelopes like $f(t)$, interchanging the roles of A and B and replacing t by $-t$. The Hausdorff-distance between A and $B + t$ is then

$$h(t) = max(f(t), g(t)).$$

Again, for L_2 the graph of h is composed of piecewise "conic" segments. We are searching for $\min_t h(t)$. This minimum is found by determining all local minima of $h(t)$. By the bounds on the number of these minima derived in [HKS93], algorithms are obtained for matching 2- and 3-dimensional finite point sets under translations minimizing the Hausdorff distance. In 2 dimensions their running times are $O(nm(n + m)\log nm)$ for the L_1- and L_∞-metric and $O(nm(n + m)\alpha(nm)\log(n + m))$ for other L_r-metrics, $r = 2, 3, \ldots$. In 3 dimensions time $O((nm)^2(n + m)^{1+\varepsilon})$ is obtained for the L_2-metric.

At first glance it is not clear how the technique described earlier can be generalized from translations to arbitrary rigid motions. However, this is done by Huttenlocher et al. in [HKK92] by considering so called *dynamic Voronoi diagrams*. Here, it is assumed that we have a point set consisting of k rigid subsets of n points each. Each of the subsets may move according to some continuous function of time. The dynamic Voronoi diagram is the subdivision of the 3-dimensional space-time such that every cross section obtained by fixing some time t equals the Voronoi diagram at time t. The authors investigate how many topological changes the Voronoi diagram can undergo as time passes which gives upper bounds on the complexity of the dynamic Voronoi diagram.
These results are applied to matching under rigid motion by representing the optimal solution as

$$D(A, B) = \min_{x,\Theta} \delta_H(r_\Theta(A), B + x)$$

where $x \in \mathbb{R}^2$ is the translation vector, and r_Θ is the rotation around the origin by angle $\Theta \in [0, 2\pi)$. For fixed Θ we have the situation described before in the case of translations. The (directed) optimal Hausdorff-distance can be determined by finding the minimum of the upper envelope of m Voronoi surfaces, namely the ones of $r_\Theta(A) - b_1, \ldots, r_\Theta(A) - b_m$. The minimum algorithm keeps track of this for changing values of Θ by considering the dynamic Voronoi diagram of these sets where Θ is identified with the time parameter. As a consequence, an optimal match of two point sets under arbitrary rigid motions can be found in time $O((m + n)^6 \log(mn))$.

Matching of point patterns under translations in *higher dimensions* is investigated by Chew et al. [CDEK95]. For the decision problem in case of the L_∞-metric, the space of feasible translations is an intersection of unions of unit boxes. This space is maintained using a modification of the data structure of *orthogonal partition trees* by Overmars and Yap [OY91]. This gives algorithms for the decision problem which are used to solve the optimization problem by parametric search [Meg83], [Col84]. In particular, for the L_∞-metric an algorithm of running time $O(n^{(4d-2)/3}\log^2 n)$ is obtained where n is the number of points in both patterns. For d-dimensional point patterns under the L_2-metric, the matching takes time $O(n^{\lceil 3d/2 \rceil+1} \log^3 n)$.

The methods described before are probably quite difficult to implement and numerically unstable due to the necessary computation of intersection points of algebraic surfaces. A much simpler but practically more promising method is given by Goodrich et al. in [GMO94]. For a "pattern" P and a "background" B of m and n points, respectively, it approximates the optimal directed Hausdorff-distance $\min_T \tilde{\delta}_H(T(P), B)$ up to some constant factor. T ranges over all possible transformations which are translations in arbitrary dimensions or rigid motions in 2 or 3 dimensions. The approximation factors are between $2+\varepsilon$ for translations in $\mathbb{R}^d$ and $8+\varepsilon$ for rigid motions in $\mathbb{R}^3$. The running times are considerably faster than the ones of algorithms computing the optimum exactly. The algorithm for rigid motions in $\mathbb{R}^2$ essentially works as follows:

1. Fix a pair (p, q) of diametrically opposing points in P.
2. Match the pair by some rigid motion as good as possible to each pair of points of B.
3. For each such rigid motion determine the distance of the image of each point in P to its nearest neighbor. Choose that match where the maximum of these distances is minimized.

In higher dimensions the nearest neighbor search is done approximately by the data structure of Arya et al. [AMN$^+$94].

2.2.3 Practical Variations

Percentile-Based Hausdorff Distance

As was mentioned above, the Hausdorff-distance is probably the most natural function for measuring the distance between sets of points. Furthermore, it can easily be applied to *partial matching* problems. In fact, suppose that sets A and B are given where A is a large "image" and B is a "model" of which we want to know whether it matches some part of A. This is the case exactly if there is some allowable transformation T such that the one-way Hausdorff-distance $\tilde{\delta}_H(T(B), A)$ is small. In fact, many of the matching algorithms with respect to Hausdorff-distance presented previously can be applied to partial matching as well. An application of this property to the matching of binary images is given by Huttenlocher et al. in [HKR93] where a discrete variant of the Voronoi-diagram approach for matching under translation with respect to Hausdorff-distance is used.

In the same article a modification of the Hausdorff-distance is suggested for the case that it is not necessary to match B completely to some part of A but only at least k of the m points in B. In fact, the distance measure being used is

$$h_k(B, A) = \min_{b \in B}{}^k \min_{a \in A} ||a - b||$$

where $\min_k$ denotes the k-th smallest rather than the largest value. This percentile definition allows us to overcome the sensitivity of the Hausdorff-distance to *outliers*, which is very important in practice.

The paper [HKR93] is also interesting in that the authors show how to adapt some of the conceptual geometric ideas presented earlier to their rasterized context so as to obtain, after several other optimizations they invented, efficient practical algorithms for the partial Hausdorff matching described above.

Alignment and Geometric Hashing

A number of other techniques for point pattern matching have been developed and used in computer vision and computational molecular biology. In computer vision the point pattern matching problem arises in the context of *model-based recognition* [EG90, Bai84] — in this case we are interested in knowing whether a known object (we will call it the model M) appears in an image of a scene (which we will denote by S). Both the model and the scene are subjected to a feature extraction process whose details need not concern us here. The outcome of this process is to represent both M and S as a collection of geometric objects, most commonly points. The same principle applies to the molecular biology applications — typically molecular docking [NFWN94]. In that context we are trying to decide if the pattern, usually a small molecule (the ligand), can sterically fit into a cavity, usually the active site of some protein. Again through a feature extraction process, both the ligand and the active site can be modeled by point sets. The dimensionalities of the point sets M and S, as well as the transformation group we are allowed to use in matching M to S, are application dependent. In computer vision S is always 2-D while M can be either 2-D or 3-D; both are 3-D in the biology context.

To illustrate the methods of alignment and of geometric hashing we use below an example in which both M and S are planar point sets of cardinalities m and s respectively. In the example we will assume that the allowed transformation group when matching M to P is the group of similarities, i.e. Euclidean transformations and scalings. We are interested in one-way matches from M to P, in the sense of the one-way percentile Hausdorff distance: we will be looking for transformations that place many (most) points of M near points of P. The extension of these ideas to the case of other dimensions and other transformation groups is in general straightforward; the one exception is when the allowed transformations include a (dimension-reducing) projection — about which more below.

In the *alignment method* [HU90], two points a and b of M are first chosen to define a reference coordinate frame $[a;b]$ for the model. We can think of the first point as the origin $(0,0)$ and the second point as the unit along the x-axis $(1,0)$. This choice also fixes the y-axis and thus an orthogonal coordinate system in which all points of M can be represented by two real values. Note that this representation of the points in M is invariant under translations, rotations, and scalings. We now *align* the points of a and b of M with two chosen points p and q of S respectively. Up to a reflection, this fixes a proposed similarity mapping M to S (we will ignore the reflection case in what follows). In order to test the goodness of this proposed transformation, we express all points of S using coordinates in the frame $[p;q]$. Now that we have a common coordinate system for two sets, we just check for every point of M to see if there is a point of S nearby (within some preselected error tolerance). The number of points on M that can thus match in this verification step is the score for the particular transformation we are considering.

The alignment method consists of trying in this way to align pairs of points of M with pairs of points of S and in the process discover those transformations that have the highest matching score. If we could assume that all points of M are present in S, in principle we could get by by matching and verifying a specific pair from M with all its counterparts in S. Because of occlusions, however, this assumption cannot be made in practice and usually we need to try many pairs of points from M before the correct match is found. Alignment is thus an exhaustive method and its worst-case combinatorial complexity is bad, $O(m^3 s^2)$ — even assuming only a linear $O(m)$ verification cost ($O(m \log s)$ would be a more theoretically correct bound). Things get worse as the size of the frames we need to consider increases with higher dimensions or larger transformation groups. Thus in the vision context a lot of attention must be given to the feature extraction process, so that only the most critical and significant features of each of M and S are used during the alignment.

Since we may want to match the same model M into many scenes or, conversely, we may be looking for the presence of any one of a set of models $M_1, M_2, \ldots, M_k$ in a given scene, it makes sense to try to speed up the matching computation through the use of preprocessing. This leads to to the idea of *geometric hashing* [LW88, LSW88a, LSW88b]. Let us describe geometric hashing in the same context as the in the above alignment problem, but with several models $M_1, M_2, \ldots, M_k$. As above, for each model M_i and each frame $[a; b]$ of two points for that model, we calculate coordinates for all other points of M_i in that frame. The novel aspect of geometric hashing is that these coordinates are then used as a key to hash into a global hash table. We record at that hash table entry the model index and frame pair the entry came from. The computation of this hash table completes the preprocessing phase of the algorithm. Note that there is a hash table entry for each triplet (model, frame for that model, other point in that model); multiple triplets may hash to the same table entry, in which case they are linked together in a standard fashion.

At recognition time, we choose a frame pair $[p; q]$ in the scene S and compute the coordinates of all points of S in that frame. Using then these coordinates as a key, we then hash into the global hash table and 'vote' for each (model, frame) pair stored at that hash table entry. If we were lucky to choose two scene points which correspond to two points $[a; b]$ in an instance of some model M_i, we can then expect that the pair $(M_i, [a; b])$ will get many votes during that process, thus signaling the presence of M_i in the scene and also indicating the matching transformation involved. In general, of course, we cannot expect to be so lucky in choosing p and q the first time around, so we will have to repeat the voting experiment several times. In the worst case, preprocessing for a model M of size m costs $O(m^3)$ and the recognition by voting also costs $O(s^3)$ (we assume throughout that the cost of accessing the hash table is constant). Note that by appropriately rounding the coordinates used as a key to the hash table we can allow for a certain error tolerance in matching points of M and S. Also, once some promising (model, frame) pairs have been identified, the votes for the winner actually give us a proposed correspondence between model and scene points. The matching transformation can then be calculated more accurately using a least-squares fit [LSW88b].

As was mentioned above for the alignment problem, these ideas also extend to matching point sets in 3-D, as well to other transformation groups, such as the group of affine maps. One noteworthy aspect of this in the vision case is the dimension-reducing projection maps that must be allowed in matching (as when M is 3-D but S is 2-D). This makes the problem harder, as the projection map is not invertible and a point in S has an inverse image which is a line in 3-D. This can be handled in geometric hashing by having each point of S, after a frame has been chosen, generate samples along a line of possible matching points from M in 3-D and vote for each of them separately [LW88].

Geometric hashing has been successfully used in both identifying CAD/CAM models in scenes, as well in the molecular docking problem [LW88, LSW88a, LSW88b, NFWN94, NLWN94]. Performance in practice tends to be much better than the above combinatorial worst-case bounds would indicate. Recent theoretical studies also suggest that randomization can improve the above bounds for alignment and geometric hashing, especially in cases where the model is not present in the the scene, or when the point sets involved have limited 'self-similarity' [IR].

3 Matching of Curves and Areas

Apart from point patterns, research has been done in recent years also on the resemblance of more complex patterns and shapes, mostly in two dimensions. These objects usually are assumed to be given by polygonal chains or one or more simple polygons representing their boundary. As a measure for their resemblance usually the Hausdorff-distance is used, though some articles are concerned with other variants, as described in Section 2.2.3.

3.1 Optimal Matching of Line Segment Patterns

Throughout this section we will assume, if not explicitly stated otherwise, that the input to the algorithms consists of two sets A, B of n, m line segments in two dimensions, respectively. The aim is to find an optimal match between A and B, i.e. a transformation T minimizing the Hausdorff-distance $\delta_H(A, T(B))$. Here, A and B are identified with the sets of points lying on their line segments and the metric underlying the Hausdorff-distance is L_2.

Notice that while there is a straightforward $O(nm)$ algorithm for *computing* the Hausdorff distance between fixed finite point sets A and B, this is no longer the case for sets of line segments. In the case of convex polygons Atallah [Ata83] gave a linear time algorithm. For arbitrary sets of line segments an asymptotically optimal $O(n \log n)$ algorithm was given by Alt et al. in [ABB91]. This algorithm is based on the fact, that the Hausdorff distance can only occur at endpoints of line segments or at intersection points of line segments of A with the Voronoi diagram of B or vice versa. Furthermore, for any Voronoi edge this can happen only at the two extreme intersection points with line segments of the other set. These points are then determined by a line sweep algorithm.

In [ABB91] it was also observed that the *matching* problem under translations or rigid motions can be solved in polynomial time. These results are based on the fact that if the transformation has k degrees of freedom (e.g. $k = 3$ for rigid motions) then in the optimal position the Hausdorff-distance essentially must occur at at least $k+1$ different places.

More sophisticated techniques leading to asymptotically faster, but probably practically quite complicated algorithms are used by Agarwal et al. [AST94] for translations and Chew et al. [CGH$^+$93] for arbitrary rigid motions. Both articles start with essentially the same idea. They first solve the decision problem whether for given A, B, and $\varepsilon > 0$ there exists a transformation T such that $\delta_H(A, T(B)) \leq \varepsilon$. Let us consider the one-way Hausdorff-distance in detail, with just a few technical details it can be extended to the two way Hausdorff distance.
Let C_ε be the disk of radius ε around the origin and A_ε the Minkowski sum $A \oplus C_\varepsilon$. Clearly, A_ε is the union of so called "racetracks" ([AST94]), i.e. rectangles of width 2ε with semidisks of radius ε attached at their ends (see Figure 3).

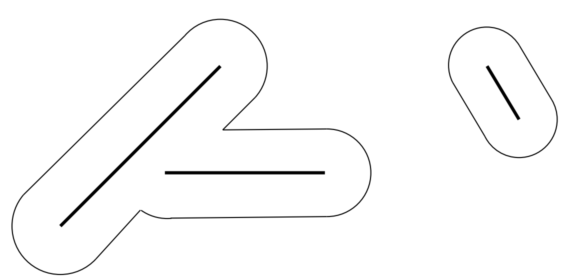


Figure 3: The set A_ε.

Now, for a transformation T the one-way Hausdorff-distance $\tilde{\delta}_H(T(B), A) \leq \varepsilon$ exactly if $T(B) \subset A_\varepsilon$. Let us consider the case of translations first. Suppose t is a translation vector not satisfying the inclusion above. So we have $B + t \not\subset A_\varepsilon$, in particular $t + b_i \not\subset A_\varepsilon$ for some line segment $b_i \in B$. This is equivalent to $t \in \overline{A}_\varepsilon \oplus (-b_i)$, where $\overline{A}_\varepsilon$ denotes the complement $I\!R^2 \setminus A_\varepsilon$ of A_ε. Conversely, a translation t moves B into A_ε exactly if $t \in \overline{\overline{A}_\varepsilon \oplus (-b_i)}$ for $i = 1, \ldots, m$. The latter set we will denote by A_i^ε, (see Figure 4) so there is a one way match exactly if the set

$$S = \bigcap_{i=1}^{m} A_i^\varepsilon$$

is nonempty.

Figure 4 shows that in the construction of the sets A_i^ε, $i = 1, \ldots, m$, we use the circular arcs and line segments bounding A_ε and, additionally, these curves translated by the vector b_i; altogether there are $O(nm)$ circular arcs and line segments. Each A_i^ε is a union of some of the cells of the *arrangement* $\mathcal{A}$ defined by these curves. The decision problem for translations can be solved by a sweep line algorithm. While sweeping across the arrangement the algorithm computes the *depth* of the cells, i.e. the number of different A_i^ε that cover a cell. Clearly, S is nonempty, exactly if there is a cell of depth m.

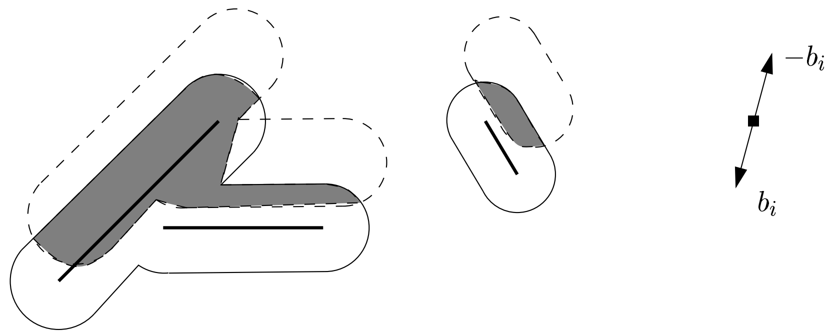


Figure 4: The set A_i^ε (shaded).

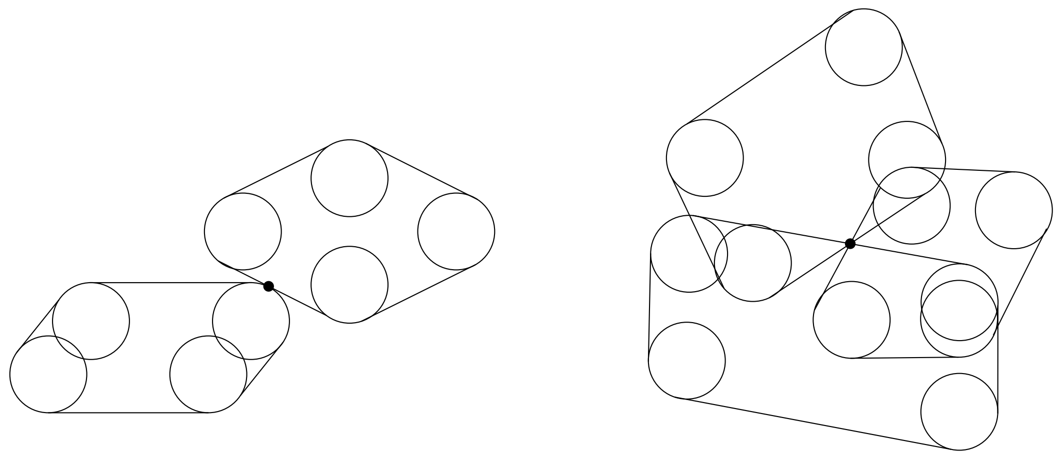
Since the complexity of the arrangement is $O((mn)^2)$ the sweep line algorithm solves the decision problem for translations in $O((mn)^2 \log(mn))$ time (see [AST94]). In the case of *rigid motions* (see [CGH$^+$93]) we assume that first the set B is rotated around the origin and then translated in order to match the set A. For each orientation $\theta \in [0, 2\pi)$ we consider the sets $A_i^\varepsilon(\theta)$ which are defined like A_i^ε, only that b_i is rotated by angle θ around the origin. Likewise the arrangement $\mathcal{A}(\theta)$ depends on the orientation θ.

$\mathcal{A}(0)$ and the depth of its cells can be determined as described before. Then while increasing θ the algorithm keeps track of the changes that occur in the arrangement and of the depth of the newly appearing cells. The arrangement only changes topologically for orientations θ where two sets of the form $(a_j \oplus \varepsilon) \oplus (-b_j)$ touch each other, a so-called "double event"(see Figure 5 a)), or the boundaries of three of them intersect in one point, a "triple event" (see Figure 5 b)).

Altogether, there are $O(m^3n^3)$ events. The algorithm determines them and sorts them in a preprocessing phase. Then it makes use of a suitable data structure where all local changes in the arrangement due to an event can be processed in constant time. In this manner, starting from $\mathcal{A}(0)$ all arrangements are inspected whether for some θ there is a cell in $\mathcal{A}(\theta)$ of depth m. If so, a positive answer is given to the decision problem. Altogether the algorithm requires $O(m^3n^3 \log(mn))$ time.

In both cases the algorithms for the decision problem can be turned into ones for the *optimization problem* by *parametric search* (see [Meg83] [Col84]) on the parameter ε. For matching by translation, Agarwal et al. [AST94] describe a parallel algorithm which first computes the arrangement $\mathcal{A}$. Then it determines the depth of its cells by considering the dual graph, finding an Eulerian path in it and using that to traverse the cells systematically. This parallel algorithm is used to direct the parametric search. Altogether, an $O((mn)^2 \log^3(mn))$ algorithm for finding the optimal matching under translations is obtained.

For optimal matching by rigid motions Chew et al. [CGH$^+$93] use an EREW-PRAM sorting algorithm for the events to direct the parametric search. Whenever this algorithm attempts to compare two events $\theta_1(\varepsilon)$, $\theta_2(\varepsilon)$, the set of critical parameters ε is determined



a) b)

Figure 5: Orientations where the arrangement changes: a) double event b) triple event.

and sorted. Then a binary search is done on these critical parameters in each step involving the decision algorithm described before to determine the interval containing the optimal ε. Altogether an $O((mn)^3 \log^2(mn))$ algorithm is obtained for optimal matching under rigid motions.

With essentially the same ideas and the usage of dynamic Voronoi diagrams efficient algorithms for point pattern matching are obtained in [CGH$^+$93], as was mentioned in section 2.

Huttenlocher et al. [HKS93] also extend the Voronoi diagram approach described in section 2 to sets of line segments. This method leads to rather complicated surfaces if the Hausdorff-distance with respect to the L_2-metric is considered. However, if the underlying metric is L_1 or L_∞ the situation is simpler and an $O((mn)^2\alpha(mn))$ algorithm can be obtained.

3.2 Approximate Matching

As we have seen in the previous sections the algorithms for finding the optimal solution of the matching problem use quite sophisticated techniques from computational geometry and, therefore, are probably too complicated to implement. Also the asymptotic running times are, although polynomial, rather high. One approach to overcome these problems are *approximation algorithms* for the optimization problem. These are algorithms that do not necessarily find the optimal solution, but one whose distance is within a constant factor c of the optimum. Again, if not explicitly stated otherwise we will consider matching of sets of line segments with respect to Hausdorff-distance based on the L_2-metric.

Approximation algorithms in this context were considered first by Alt et al. [ABB91, ABB95] using so called *reference points*. These are points r_A, r_B that are assigned to the sets A and B and have the property that when B is transformed to match A optimally then the distance of the transformed r_B to r_A is also bounded by a constant factor a times the Hausdorff-distance of the matching. a is called the *quality* of the reference point.

A reference point can be found very easily in the case of translations as allowable transformations. In fact, to a set A assign the point $r_A = (x^A_{min}, y^A_{min})$ where x^A_{min} (y^A_{min}) is the lowest x-coordinate (y-coordinate) of any point in A. So r_A is the lower left corner of the smallest axes-parallel rectangle enclosing A. Observe that if in an optimal match A and the translated image B' of B have Hausdorff-distance δ then $\left|x^A_{min} - x^{B'}_{min}\right| \leq \delta$ and $\left|y^A_{min} - y^{B'}_{min}\right| \leq \delta$ (see Figure 6). Consequently, the distance of r_A and $r_{B'}$, which is the image of r_B under the translation, is at most $\sqrt{2}\delta$. So r_A is a reference point for A of quality $\sqrt{2}$.

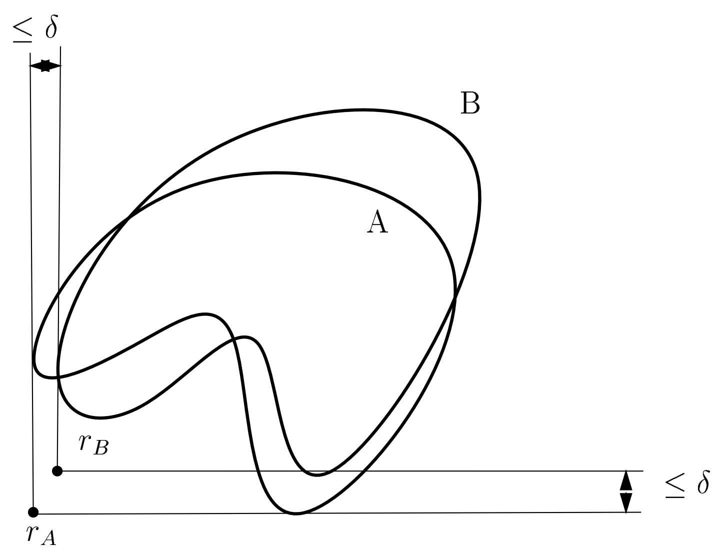


Figure 6: Optimal match between A and B under translations.

Now, suppose that instead of finding the optimal translation of B to match A we use just the one that matches r_B to r_A obtaining an image B'' of B. Then, since B' is obtained from B'' by translation by the vector $r_{B'} - r_A$, $\delta_H(B', B'') \leq \sqrt{2}\delta$. Consequently

$$\begin{aligned} \delta_H(A, B'') &\leq \delta_H(A, B') + \delta_H(B', B'') \\ &\leq (\sqrt{2} + 1)\delta. \end{aligned}$$

So the Hausdorff-distance of the match found by the reference points is at most by a factor $\sqrt{2} + 1 \approx 2.4$ worse than the optimal one. In general, matching with respect to a reference point of quality a for translations yields a match that is at most a factor $a+1$ worse than the optimal one. Observe that the approximation algorithm has linear running time, since only the reference points need to be determined. So it is much faster and much simpler than the best known algorithm for finding the optimal match which has running time $O((mn)^2 \log^3 mn)$ [AST94].

Reference points for rigid motions are not that easy to find. Obviously the one for translations given above does not work any more. Nor do the seemingly obvious choices like the center of gravity of the convex hull of a set A or the center of the smallest enclosing circle. It was shown by Alt et al. [ABB91] that for an arbitrary bounded set *A the center of gravity of the boundary of the convex hull* is a reference point with respect to rigid motions. However, the upper bound given for the quality of this reference point is rather large, in fact, it is $4\pi + 4 \approx 16.6$.

Aichholzer et al. in [AAR94] found a better reference point. In fact, imagine that the axes-parallel wedge determining the reference point for translations given above is circled around the set A. Then its apex describes a closed curve. For the "average point"on this curve no particular direction is preferred, so it might be a candidate for a reference point under rigid motions. Formalizing and slightly simplifying this idea, we obtain the following definition for arbitrary bounded sets A:

$$s(A) := \frac{1}{\pi} \int_0^{2\pi} h_A(\phi) \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} d\phi$$

where $h_A(\phi)$ is the so called *support function* of A which assigns to ϕ the largest extent of A in direction ϕ.

The point $s(A)$ is the so called *Steiner-point* of A. The Steiner point is well investigated in the field of convex geometry [She66, Grü67] as well as in functional analysis [PY89]. Using these results it is shown in [AAR94] that the Steiner point is a reference point not only for translations and rigid motions but even for *similarities* and not only for two but for arbitrary dimensions. Its quality is $4/\pi \approx 1.27$ in two, 1.5 in three and between $\sqrt{2/\pi}\sqrt{d}$ and $\sqrt{2/\pi}\sqrt{d+1}$ in d dimensions. Usually the Steiner point is defined for convex bodies only, it can be extended to arbitrary bounded sets by taking the Steiner point of the convex hull. In the case of sets of line segments we obtain convex polygons for which the Steiner point can easily be computed. In fact it is the weighted average of the vertices where each vertex is weighted by its exterior angle divided by 2π. Furthermore, Przesławski and Yost [PY89] showed that for translations there is no reference point whose quality is better than the one of the Steiner point.

In the case of translations the usage of a reference point for approximate matching was obvious. In the case of rigid motions first the two reference points are matched by a translation and then the optimal matching is sought under rotations around the common reference point. This is easier than the general optimization problem since the matching of the reference points reduces the number of degrees of freedom by two (in two dimensions). In the case of similarities, figure B is first stretched by the factor d_A/d_B, where d_A and d_B are the diameters of A and B, respectively. Then the algorithm for rigid motions is applied.

In [AAR94] it is shown that from reference points of quality a approximation algorithms are obtained yielding a solution within a factor of $a+1$ of the optimal one in the case of rigid motions and within a factor of $a+3$ of the optimal in the case of similarities. The running times for both approximate matching algorithms are $O(nm\log(nm)\log^*(nm))$.

Finally it should be mentioned that by an idea due to Schirra [Sch88] it is possible to get the approximation constant of reference point based matching with respect to translations or rigid motions arbitrarily close to 1. In fact, suppose that the quality of the reference point is a. This means that in the optimal match the distance of reference point r_B is mapped into the $a\delta$-neighborhood U of r_A. In order to achieve an approximation constant $1+\varepsilon$ for a given ε, we place onto U a sufficiently small grid so that no point in U has distance greater than $\varepsilon\delta$ from the nearest grid point. Instead of placing r_B onto r_A only we place it onto each grid point and proceed as described before. Since at some point r_B is placed at a distance of at most $\varepsilon\delta$ of its optimal position, the approximation constant is at most $1+\varepsilon$. Notice, that for a constant ε only constantly many grid points are considered so the running time only changes by a constant factor.

3.3 Distance Functions for Non-Point Objects

In some applications, the simplicity of the Hausdorff-distance can be a disadvantage. In fact, when the distance between *curves* is measured the Hausdorff-distance may give a wrong picture. Figure 7 shows an example where two curves have a small Hausdorff-distance although they have no resemblance at all.

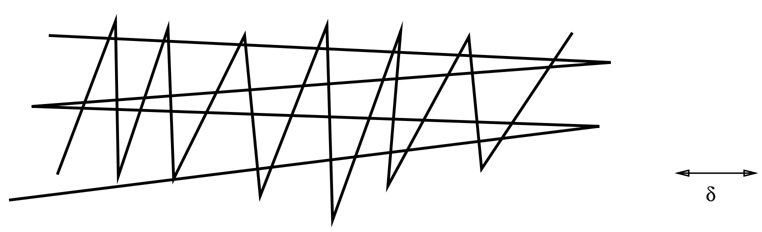


Figure 7: Two curves with small Hausdorff-distance δ.

The reason for this problem is that the Hausdorff-distance is only concerned with the point sets but not with the course of the curves. A distance considering the curves' courses can informally be illustrated as follows: Suppose a man is walking his dog, he is walking on one curve, the dog on the other. Both are allowed to control their speed but not to go backward. What is the shortest length of a leash that is possible? Formally, this distance measure between two curves in d-dimensional space can be described as follows

$$\delta_F(f,g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} ||f(\alpha(t)) - g(\beta(t))||$$

where $f,g : [0,1] \to \mathbb{R}^d$ are *parameterizations* of the two curves and $\alpha,\beta : [0,1] \to [0,1]$ range over all continuous and monotone increasing functions. This distance measure is known under the name *Fréchet-distance.*

The Fréchet-distance seems considerably more difficult to handle than the Hausdorff-distance. No matching algorithms have been developed yet, the following algorithm for measuring the Fréchet-distance between two polygonal chains has been given by Alt and Godau [AG92, AG95].

Let P and Q be the given polygonal chains consisting of n and m line segments respectively. First we consider the decision problem, so in addition to P and Q some $\varepsilon > 0$ is given and we want to decide whether $\delta_F(P,Q) \leq \varepsilon$. We first consider the $m \times n$-diagram $D_\varepsilon(P,Q)$ shown in Figure 8 which indicates by the white area for which points $p \in P$, $q \in Q$ $||p-q|| \leq \varepsilon$. The horizontal direction of the diagram corresponds to the natural parameterization of P and the vertical one to that of Q. One square cell of the diagram corresponds to a pair of edges one from P and one from Q and can easily be computed since it is the intersection of the bounding square with an ellipse.

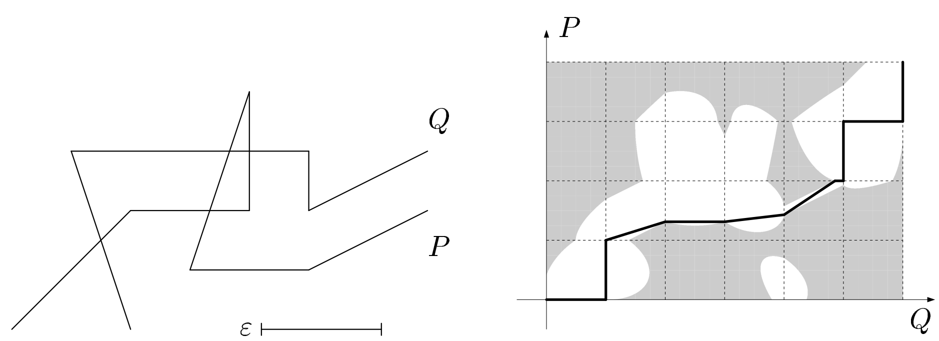


Figure 8: P, Q, ε and the diagram $D_\varepsilon(P,Q)$.

Now it follows from the definition that $\delta_F(P,Q) \leq \varepsilon$ exactly if there is a monotone increasing curve from the lower left to the upper right corner of the diagram. These considerations lead to an algorithm of running time $O(mn)$ for the decision problem. Then Cole's variant of *parametric search* [Col87] can be used to obtain an algorithm of running time $O(mn\log(mn))$ to compute the Fréchet-distance between P and Q. In practice, it seems more reasonable to determine $\delta_F(P,Q)$ bit by bit using binary search where in each step the algorithm for the decision problem is applied.

Arkin et al. [ACH+91] consider a distance function between shapes in two dimensions that is based on the slope of the bounding curves. In fact, for a given shape A some starting point O on the bounding curve is chosen and the curve is then parametrized with the natural parameterization n_A (i.e. parameterization by arc length) which is normalized so that the parameter interval is [0,1]. To each parameter $t \in [0,1]$ the angle $\Theta_A(t)$ between the counterclockwise tangent of A in the point $n_A(t)$ and a horizontal line is assigned. Θ_A is called the *turning function* of A. Θ_A is piecewise constant for simple polygons (see Figure 9), it is invariant under translations and, because of the normalization of the parameterization under scaling. A rotation of A corresponds simply to a shift in Θ–direction and a change of the origin to a shift in t–direction.

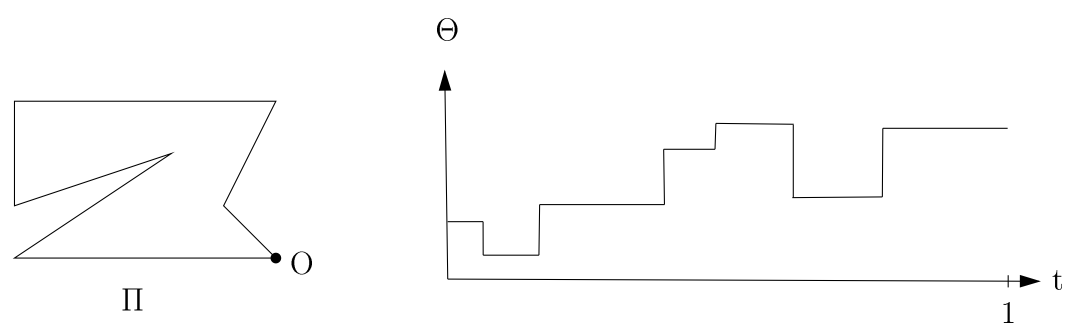


Figure 9: Turning function of a simple polygon.

Now, as a distance measure between shapes A and B the L_p–metric ($p \in I\!N$) between Θ_A and Θ_B is being used, i.e. we define

$$\delta_p(A,B) = \left(\int_0^1 |\Theta_A(s) - \Theta_B(s)|^p ds \right)^{\frac{1}{p}}.$$

Then this distance measure is made invariant under translations and the choice of the starting point 0:

$$d_p(A,B) = \min_{\Theta \in I\!R, t \in [0,1]} \left(\int_0^1 |\Theta_A(s+t) - \Theta_B(s) + \Theta|^p ds \right)^{\frac{1}{p}}.$$

It is shown that d_p is a metric for all $p \in I\!N$.

An algorithm is given for *computing* $d_2(A,B)$ where A, B are simple polygons with n and m edges, respectively. It can be shown that the minimum in the definition of d_p can occur at at most $O(mn)$ "critical" values of t. By considering partial derivatives with respect to Θ it is shown that for any fixed t the optimal Θ can easily be computed. Altogether, an $O(mn \log mn)$ algorithm is obtained. An extension of this algorithm to deal with scaling as well was recently given by Cohen and Guibas [CG97].

The drawback of this distance measure is its sensitivity to noise, especially non-uniformly distributed noise, but it works properly if the curves are sufficiently smooth.

A generalization of this distance to a distance measure that is invariant under affine transformations is given by Huttenlocher and Kedem [HK90]. With respect to this distance, matching of two polygons with n and m vertices under affine transformations is possible in time $O(mn \log mn)$.

A different idea is to represent shapes by their areas rather than by their boundaries. In this context the probably most natural distance measure between two shapes is the *area* *of their symmetric difference*. However, this measure seems to be much more difficult to handle than Hausdorff–distance. Within computational geometry it was first considered in a paper by Alt et al. [ABGW90] in connection with the very special problem of optimally approximating a convex polygon by an axes–parallel rectangle. Only recently some more results on the symmetric difference have been obtained. In fact, de Berg et al. in [dBDvK$^+$96] consider matching algorithms maximizing the area of overlap of convex polygons which is the same as minimizing the symmetric difference. They obtain an algorithm of running time $O((n+m)\log(n+m))$ for translations where n and m are the numbers of vertices of the two polygons. In addition, it is shown that if just the two centers of gravity are being matched by a translation this yields a position of two convex figures where the area of overlap is within a constant factor of the maximal one. A lower bound of $9/25$ and an upper bound of $4/9$ is obtained for this constant, so the center of gravity is a *reference point* with respect to maximizing the overlap under translations.

As easily can be seen this does not imply directly that it is a reference point with respect to the area of the symmetric difference. However, this can be shown, as well. In fact, Alt et al. [AFRW96] show that if the centers of gravity of two convex figures are matched the area of the symmetric difference is at most $11/3$ times the minimal one. It is demonstrated with an example that this bound is tight. The center of gravity is also a reference point for other sets of transformations such as rigid motions, homotheties, similarities, and arbitrary affine mappings.

4 Shape Simplification and Approximation

In manipulating shape representations, it is often advantageous to reduce the complexity of the representation as much as possible, while still staying geometrically close to the original data. There is a vast literature on shape simplification and approximation, both within computational geometry and in the various applied disciplines where shape representation and manipulation issues arise. In this section we cannot possibly attempt to survey all the approaches taken and the results obtained. We will focus on a small subset of results giving efficient algorithms for shape simplification under precise measures of the approximation involved.

4.1 Two-dimensional Results

To focus our attention, let us consider the problem of simplifying a shape represented as a polygonal chain in the plane. Our goal is to find another polygonal chain with a smaller number of links and which stays close to the original. Let the original chain be C, defined by n vertices $v_1, v_2, \ldots, v_n$ (we assume in this discussion that C is open), and let the desired approximating chain A be defined by m vertices $w_1, w_2, \ldots, w_m$. A number of variations on the problem are possible, depending on exactly how the error, or distance, between C and A is measured, and on whether the w_j's need to be a subset of the v_i's, or can be arbitrary points of the plane. In general we have to solve one of two problems in finding A:

The min-# problem: Find an A which minimizes m (the combinatorial complexity of A), given some *a priori* bound ϵ on the maximum allowed distance between C and A, or

The min-ϵ problem: For a given m, find an A which minimizes the distance ϵ between C and A.

In order to illustrate some of the ideas involved in solving these problems, let us further assume that both C and A are x-monotone chains — in other words, C and A are piecewise linear continuous functions $C(x)$ and $A(x)$. For such chains a very natural measure of distance is the so-called *uniform* or *Chebychev* metric defined as

$$d(A, C) = \max |A(x) - C(x)|,$$

where the max is taken over the support on the x-axis for C. Let us consider the case when the vertices of A can be arbitrary points, i.e. need not be vertices of C. Imai and Iri [II86, II88], and Hakimi and Schmeichel [HS91] gave optimal $O(n)$ algorithms for the min-# problem in this context. Their methods can be viewed as an extension of the linear-time algorithm of Suri [Sur86] for computing a minimum link path between two given points inside a simple polygon (here the polygon is the chain C appropriately 'fattened' vertically by the allowed error ϵ) and make crucial use of the concept of *weak-visibility* from an edge inside a simple polygon [AT81]. Similar ideas were also used by Aggarwal, Booth, O'Rourke, Suri, and Yap [ABO$^+$89] to give an $O(n \log k)$ algorithm for finding a convex polygon with the smallest number k of sides nested between two given convex polygons of total complexity n.

For the min-ϵ variant of the chain simplification problem, Hakimi and Schmeichel gave an $O(n^2 \log n)$ algorithm by cleverly limiting the critical values of ϵ to a set of size $O(n^2)$ and then using a certain kind of binary search. More recently, Goodrich [Goo94] used a number of new geometric insights to reduce the set of critical ϵ values to $O(n)$ and thus obtained an $O(n \log n)$ algorithm through several applications of pipelined parametric searching techniques. For a survey of results when A and C are not constrained to be x-monotone, see the paper of Eu and Toussaint [ET94] and related work by Hershberger and Snoeyink [HS94], and Guibas, Hershberger, Mitchell, and Snoeyink [GHMS93]. In general, algorithms for the min-# problem have linear complexity, while those for the min-ϵ have quadratic complexity.

Next let us consider the problem variant where the vertices of A have to be a subset of the vertices of C. Now we do not require that A or C be x-monotone (so we revert to using the Hausdorff distance function). One of the oldest and most popular algorithms for this problem is the heuristic Douglas-Peucker [DP73] line simplification algorithm from the Geographical Information Systems community; Hershberger and Snoeyink showed how to implement this algorithm to run in $O(n \log n)$ time [HS92b]. A more formal approach to the problem was initiated in the papers by Imai and Iri cited above. For the min-# variant, Imai and Iri reduce the problem to a graph-theoretic problem as follows. A line segment $\overline{v_i v_j}$ joining vertices v_i and v_j of C is called a *shortcut* for the subchain $v_i, v_{i+1}, \ldots, v_j$ of C. A shortcut is *allowed* if the error it induces is at most the prescribed error ϵ; the error of the shortcut $\overline{v_i v_j}$ is defined to be the maximum distance from $\overline{v_i v_j}$ to a point v_k, where $i \leq k \leq j$. It is easy to see that this is also the Hausdorff distance from $\overline{v_i v_j}$ to the subchain $v_i, v_{i+1}, \ldots, v_j$. Our goal is to replace C by a chain consisting of allowed shortcuts.

So we consider a directed acyclic graph G whose nodes V are the vertices $v_1, v_2, \ldots, v_n$ of C and whose edges E are the pairs (v_i, v_j) if and only if $i < j$ and the shortcut $\overline{v_i v_j}$ is allowed. A shortest (in terms of the number of edges) path from v_1 to v_n corresponds to a minimum vertex simplification of C; such a path can be found in time linear in the size of G by topological sorting [CLR90]. The size of G is $O(n^2)$ and it can be computed by an obvious method in $O(n^3)$ time. Thus constructing G is the bottleneck in the computation. Melkman and O'Rourke [MO88] showed how to reduce the construction time to $O(n^2 \log n)$, and Chan and Chin [CC92] further reduced it to optimal $O(n^2)$.

The Chan and Chin algorithm starts from the observation that the error of a shortcut $\overline{v_i v_j}$, $i < j$, is the maximum of the errors of two half-lines: the one starting at v_i and going towards v_j (call it l_{ij}), and of the one starting at v_j and going towards v_i (call it l_{ji}). To compute the graph G we intersect the graphs G_1 and G_2, where G_1 contains the edge (v_i, v_j) if and only if $i < j$ and the error of l_{ij} is less than ϵ and G_2 contains the edge (v_i, v_j) if and only if $i < j$ and the error of l_{ji} is less than ϵ. We show how to compute G_1 in $O(n^2)$ time; the computation of G_2 is entirely symmetric by reversing the number of the vertices of C.

We examine the vertices of G_1 in the sequence $v_1, v_2, \ldots, v_n$. When we process vertex v_i, we calculate in turn the errors determined by all half-lines l_{ij}, where j takes on the values $i+1, i+2, \ldots, n$. Let D_k denote a closed disk of radius ϵ centered at v_k. The error of l_{ij} is at most ϵ if and only if l_{ij} intersects all disks D_k with $i \leq k \leq j$. Thus the algorithm works by maintaining the cone of half-lines from v_i which intersect the disks $D_{i+1}, D_{i+2}, \ldots, D_j$ (which is nothing but the intersection for the corresponding cones for all these disks separately). When we process v_{j+1} it suffices to update this cone, which is a constant time computation. If the cone stays non-empty, then (v_i, v_{j+1}) is in G_1; otherwise we are done with v_i as all further half-lines will also have error which is too large. Thus the computation of G_1, and therefore of G and of our desired shortest path can be done in $O(n^2)$ time. Figure 10 illustrates the situation with the disks and the cone of v_i a some intermediate point.

Methods based on the Imai–Iri graph construction seem inherently quadratic as, if ϵ is large enough, the graph will have $\Omega(n^2)$ allowed shortcuts. Very recently, Varadarajan [Var96] was able to use graph clique compression techniques such as those proposed by Feder and Motwani [FM91] to obtain an $O(n^{4/3+\delta})$ algorithm for this min-# problem in the case of x-monotone chains. Varadarajan gave a randomized and a more complex deterministic algorithm for the min-ϵ version of this problem as well, with the same time bound.

In the general (non-x-monotone) case of the above chain simplification problems it is quite possible that A may end up being self-intersecting, even though C itself is simple. This is clearly undesirable in many application contexts. Even worse, one is often simplifying several chains at once, as in the case of boundaries between regions in, say, a geographical map. In this case it is important that the topological structure of the regions be maintained after simplification, so the simplifications of disjoint chains are not allowed to end up crossing

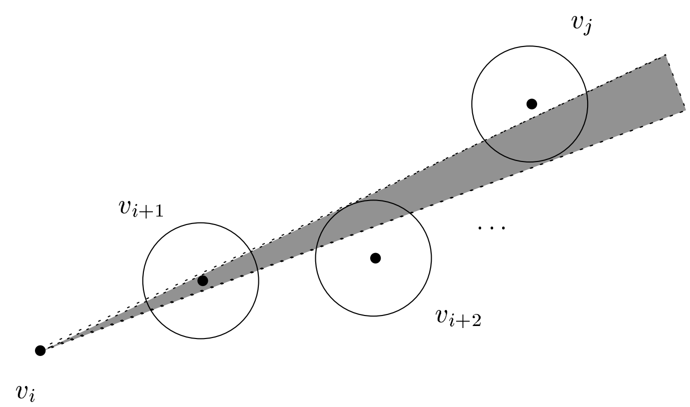


Figure 10: The computation of the allowed shortcuts starting at v_i

each other. Guibas, Hershberger, Mitchell, and Snoeyink [GHMS93] showed that the min-# problem is NP-hard in this case, when the positions of the approximating vertices can be arbitrary. When the vertices of the approximating chain have to be a subset of the original chain and we are in the x-monotone setting, de Berg, van Kreveld, and Schirra [dBvKS95] gave an $O(n(n+m)\log n)$ algorithm for the min-# problem for a chain C of n vertices so that the resulting approximating chain A is guaranteed to be simple and to be on the same side of each of m given points as C is. Using this algorithm as a local subroutine, they give a method for polygonal subdivision simplification which is guaranteed to avoid topological inconsistencies (but which need not be globally optimal).

4.2 Three dimensions

Unfortunately the situation is not an equally happy one in three dimensions. Nearly all natural extensions of the above problems are NP-hard, so most of the extant work to-date has focussed on approximation algorithms.

The analog of an x-monotone chain in 3–D is that of a *polyhedral terrain*, which is just a continuous bivariate (and possibly non-total) function $z = T(x, y)$ which happens to be piecewise linear. The complexity of a terrain can be measured by its total number of vertices, edges, and faces. The numbers of these are linearly related by Euler's relation, so most often we will use the number of faces as our measure of the complexity of a terrain. There is a plethora of techniques in the literature for simplifying polyhedral terrains, by effectively deleting vertices which lie in relatively flat neighborhoods (and retriangulating the hole thus created). Unfortunately not much has been proved about such methods. In fact, Agarwal and Suri [AS94] have shown that even the simpler problem of deciding the approximability within a vertical tolerance ϵ of a collection of n isolated points by a polyhedral terrain with at most k faces is NP-hard. Similarly, though more surprisingly, Das and Joseph [DJ90] showed that finding a convex polytope of at most k facets nested between two other convex polytopes P and Q with a total of n facets is also NP-hard, thus settling an old question first posed by Klee.

With these results in sight, researchers turned their effort to approximation algorithms. Mitchell and Suri [MS92] formalized the nested convex polytope problem as a set-cover problem by considering the regions defined on the outer polytope by tangent planes to the inner polytope. They showed that the greedy method of set covering computes a nested polytope with $O(\kappa \log n)$ facets, where κ is the complexity of an optimal nested polytope. The same approach also works for the approximation by a convex surface of n points themselves sampled from another convex surface. These algorithms run in $O(n^2)$ time. These results were extended by Clarkson [Cla93], who gave a randomized algorithm with expected time complexity $O(\kappa n^{1+\delta})$ for computing a nested polytope of size $O(\kappa \log \kappa)$. Brönnimann and Goodrich [BG94] further improved the set cover algorithm using VC-dimension ideas to obtain a deterministic algorithm that computes a nested polytope which is to within a constant factor of the optimal one.

The set cover formulation, unfortunately, does not work for the terrain approximation problem, as we cannot independently select faces in the approximating surface. Agarwal and Suri in the paper cited above formulate instead the terrain fitting problem as a geometric partitioning problem: first we project all the points to be approximated on the xy-plane; then we seek a disjoint collection of triangles in the xy-plane which cover all these points and such that each triangle satisfies a certain legality constraint w.r.t. the points it covers. This constraint, which can be formulated as a linear program, is that the triangle can be lifted to 3–D so that it ϵ-approximates all the points it contains. Agarwal and Suri gave an approximation algorithm which solves this problem and produces a covering set of legitimate triangles whose size is $O(\kappa \log \kappa)$, where again κ is the minimum number of triangles possible. Unfortunately their algorithm has a very high complexity $O(n^8)$.

5 Shape Interpolation

Shape interpolation, more commonly known as *morphing*, has recently become a topic of active research interest in computer graphics, computer animation and entertainment, solid reconstruction, and image compression. The general problem is that of continuously transforming one geometric shape into another in a way that makes apparent the salient similarities between the two shapes and 'smoothes over' their dissimilarities. Some of the relevant graphics papers are [KR91, SG92, KCP92, SGWM93]. The morphing transformation may be thought of as taking place either in the time domain, as in animation, or in the space domain, as in surface reconstruction from planar sections. The latter area has already an extensive literature of its own [FKU77], and computational geometric techniques have been used with good results [WW93, BS94] (though the problem is not always solvable [GOS96]). In general there are numerous ways to interpolate between two shapes and little has been said about criteria for comparing the quality of different morphs and notions of optimality. Often, the class of allowable or desirable morphing transforms in only vaguely specified, with the result that the problem ends up being either under- or over-constrained.

In this section we will survey the rather small amount of work in this area which relates to Computational Geometry. Let P and Q be the two shapes we wish to morph. For now we do not specify how the shapes P and Q are described, or what is the ambient space (2–D, 3–D, etc.). Most morphing algorithms operate according to the following paradigm: firstly relevant 'features' of P and Q are identified and matched pairwise; secondly, smooth motions are planned that will bring into alignment these pairs of corresponding features; and thirdly the whole morphing transformation is generated, while respecting other global constraints which the shapes P, Q, and their interpolants must satisfy. This paradigm works well when the class of shapes to be morphed consists of fairly similar objects. In cases, however, when we want to be able to morph a great variety of different shapes, the above process is commonly subdivided into a series of stages. The shapes P and Q are first 'canonicalized' by mapping them into their canonical forms $\kappa(P)$ and $\kappa(Q)$ respectively – these canonical forms are more standardized and therefore easier to morph to each other. The whole transformation is then done by going from P to $\kappa(P)$ to $\kappa(Q)$ to Q.

As the above description makes clear, there are numerous connections between the problems of shape matching and shape interpolation, and several of the matching techniques already discussed are applicable to the morphing problem, especially in the feature matching stage. It is not so obvious, but it is equally true that morphing techniques can also be used for shape comparison problems. In a certain sense, the optimum morph from P to Q is the transformation that distorts P as little as possible in order to make it look like Q. In a morphing algorithm we can assign a notion of 'work' or cost to the distortions the algorithm needs to perform. The minimum work then required to morph P into Q can then serve as a measure of the distance from shape P to shape Q. Note that such a distance function based on morphing clearly satisfies the triangle inequality.

To make these matters concrete, let us discuss a few simple examples in 2–D and 3–D. Let P be an open simple polygonal chain of m vertices $P = p_1 p_2 \dots p_m$ and Q be an open simple polygonal chain of n vertices $Q = q_1 q_2 \dots q_n$. This problem was considered by Sederberg and Greenwood [SG92]. According to our paradigm above, we first need to establish a correspondence between the 'features' of P and Q — for polygonal chains the natural notion is that of vertices (though other choices also make sense in applications). Now since m might be different from n, in general this will have to be a many-to-one, or one-to-many mapping. But where should these duplicate vertices be added?

We can represent all possible ways to pair up vertices of P with vertices of Q in sequence by considering monotone paths in the $[1..m] \times [1..n]$ grid. An example is shown in figure 11, which shows a particular matching between a chain P of 8 vertices and a chain Q of 10 vertices. A diagonal move on the monotone path corresponds to advancing on both P and Q, while a horizontal move corresponds to advancing on Q only (and thus duplicating the corresponding vertex of P.

We can choose an optimum correspondence, by selecting the path π to be of minimal cost in some appropriate sense. For example, we may want to minimize the sum of the distances of all the corresponding pairs of vertices. Sederberg and Greenwood developed a physics-based measure of the energy required to stretch and bend P into Q once the correspondence by π is given. The optimal π under such measures can be computed by classical dynamic programming techniques [CLR90] in time $O(mn)$. Similar ideas have

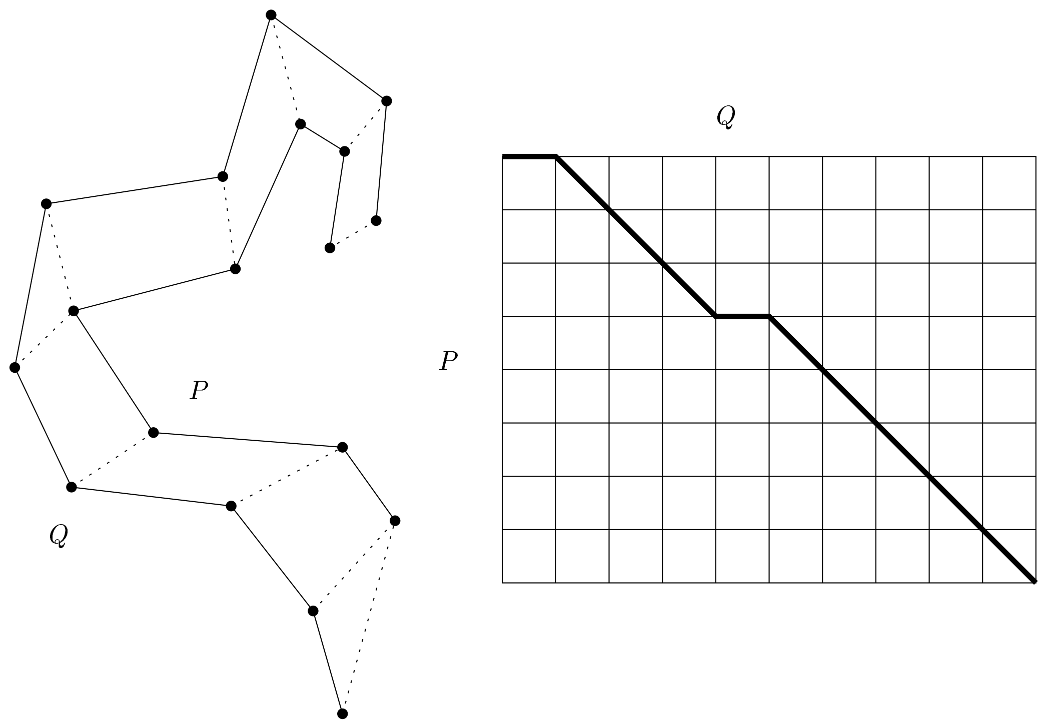


Figure 11: An example of matching polygonal chain vertices

been used to fit polyhedral sleeves to polygonal slices in parallel planes [BS94].

Once we have the correspondence, we can then move corresponding vertices to each other through linear interpolation. Implicitly, at each time t, this defines an interpolating polygonal chain R_t and thus our construction of a morph between the polygonal chains is complete. Note also that in order to extend this method to closed polygonal chains we must decide first on an appropriate 'origin' for each chain, and this is not a trivial matter. Figure 12 shows some successful and unsuccessful examples of this method, depending on the origin chosen. Note in particular that the interpolating chain R_t can self-intersect, even though neither P and Q do.

Sederberg et. al. also proposed another simple method for polygon interpolation based on interpolating side lengths and angles, once a vertex correspondence is established [SGWM93] — but now the challenge becomes to get the polygons to 'close up.'.

Preserving structural properties during a morph, such as the simplicity of a chain in the example above, is a difficult problem. Guibas and Hershberger [GH94], consider how to morph two parallel polygons to each other while maintaining simplicity. The setting is now that P and Q are two simple polygons in the plane of n sides each, and there is a 1-1 correspondence between the sides of P and Q so that corresponding sides are parallel. The goal is to continuously deform P to Q while at all times the interpolating polygon R_t has its corresponding sides parallel to those of P and Q and stays simple. In this case the very

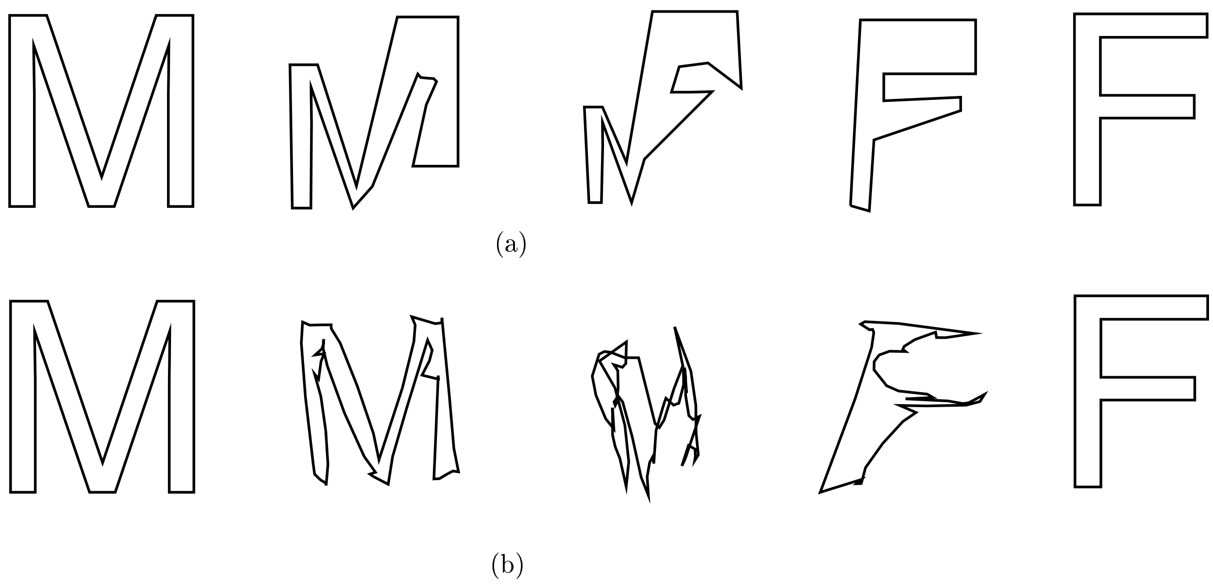


Figure 12: Examples of polygonal chain morphs: (a) a good case, (b) a bad case

statement of the problem provides the correspondence between features of the polygons. Even so, the two polygons P and Q can look quite different and the existence of a morph which remains parallel and simple is not obvious; see figure 13 (a morph between these two spiraling polygons can happen by simulating the way recording tape can move from one reel to another).

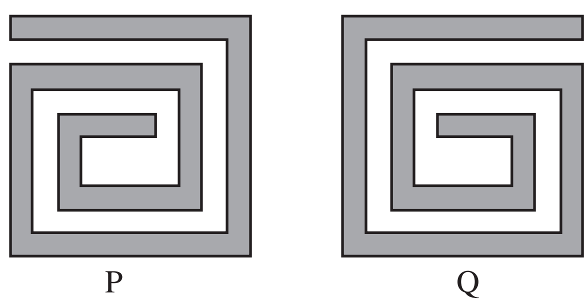


Figure 13: These oppositely spiraling parallel polygons are still morphable

Guibas and Hershberger showed that this is, nevertheless, always possible and gave an algorithm which uses $O(n^{4/3+\epsilon})$ primitive operations called 'parallel moves'; this was later improved to $O(n \log n)$ by Hershberger and Suri [HS95]. A parallel move is a translation of a side of a polygon parallel to itself, with appropriate modifications of the polygon at the endpoints of the edge. Guibas and Hershberger first showed that parallel moves can be used to take each polygon to a fractal-like canonical or reduced form in which portions of the polygon's boundary have been shrunk to micro-structures of widely different scales. A polygon in this canonical form corresponds, roughly speaking, to a binary tree whose leaf weights are the angles of the original polygon. Once P and Q are in this canonical form, the corresponding trees can be morphed into each other through a series of standard tree rotation transformations; certain validity conditions have to hold throughout this process. These tree rotations can be realized geometrically through parallel moves on the polygons. The fractal-like structure of the canonical form helps in arguing that the translations required to implement particular rotations do not interfere with other parts of the polygon.

Clearly the Guibas/Hershberger morph solves only a limited problem and even for that the canonical form used introduces unnecessarily large distortions into the interpolating shapes. Different polygon morphing techniques were developed by Shapira and Rappoport [SR95], based on the star-skeleton representation of a polygon (a decomposition of the polygon into star-shaped pieces). Such methods do much better in preserving the metric properties of the polygons, but unfortunately they still do not preserve global constraints, such as simplicity — plus they are expensive, requiring $O(n^4)$ time. Another idea for morphing polygons can be based on the compatible triangulations result of Aronov, Seidel, and Souvaine [ASS93]. They showed that P and Q can always be 'compatibly'-triangulated by adding $O(n^2)$ Steiner points (compatibility means that the triangulations are combinatorially equivalent). The use of conformal mappings has also been suggested.

Let us now also look at some work in three dimensions. The only case that has been extensively studied is that of morphing convex polytopes [KCP92]. If P and Q are convex polyhedra, a natural way to construct a matching between their surfaces is to match points on the two polyhedra that admit of the same (outwards) normal. In general, this will match all points on each face of P to a vertex of Q and vice versa, as well as matching (the points of) certain pairs of edges, one from P and one from Q. If we place the origin at an arbitrary point of space and compute the vector sums of corresponding pairs of points from P and Q, the resulting set of points will form the boundary of another convex polytope, called the *Minkowski sum* of P and Q and denoted by $P \oplus Q$ [Lat91, Lyu66]. Armed with this concept, we can then morph P to Q by constructing the *mixed volume* $(1-t)P \oplus tQ$, as t varies in the range $0 \le t \le 1$. This type of morph was exploited by Kaul and Rossignac [KR91, RK94]. The same technique works, of course, in 2–D or dimensions higher than three. A nice way to visualize this morph in 2–D is to think of P and Q as two convex polygons placed on parallel planes in 3–D. One then constructs the convex hull of the union of P and Q by adding a 'sleeve' wrapping around P and Q. The sections of this sleeve by a plane moving parallel to itself from that containing P to that containing Q gives us the morph.

The 'kinetic framework' of [GRS83] allows the extension of this type of morph to general polygons in the plane. Also, since regular subdivisions of the plane or alpha shapes [EM94] can be viewed as projections of convex polytopes in one dimension higher, the above method also gives us some possibilities for morphing such subdivisions or alpha shapes. Other approaches to morphing 2–D or 3–D shapes are given in [Gla, KCP92, Ede95].

References

[AAR94] H. Alt, O. Aichholzer, and G. Rote. Matching shapes with a reference point. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 85–92, 1994.

[ABB91] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 186–193, 1991.

[ABB95] H. Alt, B. Behrends, and J. Bloemer. Approximate matching of polygonal shapes. *Ann. Math. Artif. Intell.*, 13:251–266, 1995.

[ABGW90] H. Alt, J. Blömer, M. Godau, and H. Wagener. Approximation of convex polygons. In *Proc. 17th Internat. Colloq. Automata Lang. Program.*, volume 443 of *Lecture Notes in Computer Science*, pages 703–716. Springer-Verlag, 1990.

[ABO+89] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C. K. Yap. Finding minimal convex nested polygons. *Inform. Comput.*, 83(1):98–110, October 1989.

[ACH+91] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(3):209–216, 1991.

[AFRW96] H. Alt, U. Fuchs, G. Rote, and G. Weber. Matching convex shapes with respect to the symmetric difference. In *Proc. 4th Annual European Symp. on Algorithms- ESA'96, Springer Lecture notes in Computer science*, volume 1136, pages 320–333, 1996.

[AG92] H. Alt and M. Godau. Measuring the resemblance of polygonal curves. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 102–109, 1992.

[AG95] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[AKM+92] E. M. Arkin, K. Kedem, J. S. B. Mitchell, J. Sprinzak, and M. Werman. Matching points into pairwise-disjoint noise regions: combinatorial bounds and algorithms. *ORSA J. Comput.*, 4(4):375–386, 1992.

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

[AMWW88] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom.*, 3:237–256, 1988.

[AS94] Pankaj K. Agarwal and Subhash Suri. Surface approximation and geometric partitions. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 24–33, 1994.

[ASS93] B. Aronov, R. Seidel, and D. Souvaine. On compatible triangulations of simple polygons. *Comput. Geom. Theory Appl.*, 3(1):27–35, 1993.

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

[AT81] D. Avis and G. T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Trans. Comput.*, C-30:910–1014, 1981.

[Ata83] M. J. Atallah. A linear time algorithm for the Hausdorff distance between convex polygons. *Inform. Process. Lett.*, 17:207–209, 1983.

[Atk87] M. D. Atkinson. An optimal algorithm for geometrical congruence. *J. Algorithms*, 8:159–172, 1987.

[Bai84] H. S. Baird. *Model-Based Image Matching Using Location*. Distinguished Dissertation Series. MIT Press, 1984.

[Beh90] B. Behrends. Algorithmen zur Erkennung der ε-Kongruenz von Punktmengen und Polygonen. M.S. thesis, Freie Univ. Berlin, Institute for Computer Science, 1990.

[BG94] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 293–302, 1994.

[BS94] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 93–102, 1994.

[CC92] W. S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. In *Proc. 3rd Annu. Internat. Sympos. Algorithms Comput. (ISAAC '92)*, volume 650 of *Lecture Notes in Computer Science*, pages 378–387. Springer-Verlag, 1992.

[CDEK95] L. P. Chew, D. Dor, A. Efrat, and K. Kedem. Geometric pattern matching in d-dimensional space. In *Proc. 2nd Annu. European Sympos. Algorithms*, volume ?? of *Lecture Notes in Computer Science*, page to appear. Springer-Verlag, 1995.

[CG97] S. D. Cohen and L. J. Guibas. Partial matching of planar polylines under similarity transformations. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, page to appear, 1997.

[CGH+93] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 151–156, Waterloo, Canada, 1993.

[Cla93] Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes in Computer Science*, pages 246–252, 1993.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, Mass., 1990.

[Col84] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. In *Proc. 25th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 255–260, 1984.

[Col87] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[dBDvK+96] M. de Berg, O. Devillers, M. van Kreveld, O. Schwarzkopf, and M. Teillaud. Computing the maximum overlap of two convex polygons under translations. Technical Report, Dept. of Comp. Science, Univ. of Utrecht, 1996.

[dBvKS95] M. de Berg, M. van Kreveld, and S. Schirra. A new approach to subdivision simplification. In *Proc. of Auto-Carto 12*, pages 79–88, 1995.

[DJ90] G. Das and D. Joseph. The complexity of minimum convex nested polyhedra. In *Proc. 2nd Canad. Conf. Comput. Geom.*, pages 296–301, 1990.

[DP73] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10(2):112–122, December 1973.

[Ede95] H. Edelsbrunner. A combinatorial approach to cartograms. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 98–108, 1995.

[EG90] W. Eric and L. Grimson. *Object Recognition by Computer: the Role of Geometric Constraints.* MIT Press, 1990.

[EI96] Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 301–310, 1996.

[EM94] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, January 1994.

[ET94] D. Eu and G. Toussaint. On approximating polygonal curves in two and three dimensions. *Comput. Vision, Graphics, & Image Processing*, 56:231–246, 1994.

[FKU77] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Commun. ACM*, 20:693–702, 1977.

[FM91] T. Feder and R. Motwani. Clique partitions, graph compression, and speeding up algorithms. In *Proc. 23rd ACM Symp. Theory of Computing*, pages 123–133, 1991.

[GH94] L. Guibas and J. Hershberger. Morphing simple polygons. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 267–276, 1994.

[GHMS93] L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Internat. J. Comput. Geom. Appl.*, 3(4):383–415, December 1993.

[Gla] A. Glassner. Metamorphosis of polyhedra. Manuscript. (1991).

[GMO94] M. T. Goodrich, J. S. Mitchell, and M. W. Orletsky. Practical methods for approximate geometric pattern matching under rigid motion. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 103–112, 1994.

[Goo94] M. T. Goodrich. Efficient piecewise-linear function approximation using the uniform metric. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 322–331, 1994.

[GOS96] C. Gitlin, J. O'Rourke, and V. Subramanian. On reconstructing polyhedra from parallel slices. *Internat. J. Comput. Geom. Appl.*, 6(1):103–122, 1996.

[GRS83] L. J. Guibas, L. Ramshaw, and J. Stolfi. A kinetic framework for computational geometry. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 100–111, 1983.

[Grü67] B. Grünbaum. *Convex Polytopes.* Wiley, New York, NY, 1967.

[HK90] D. P. Huttenlocher and K. Kedem. Computing the minimum Hausdorff distance for point sets under translation. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, 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 *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 110–120, 1992.

[HKR93] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15:850–863, 1993.

[HKS93] D. P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete Comput. Geom.*, 9:267–291, 1993.

[HS91] S. L. Hakimi and E. F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graph. Models Image Process.*, 53(2):132–136, 1991.

[HS92a] P. J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 93–101, 1992.

[HS92b] J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line simplification algorithm. In *Proc. 5th Intl. Symp. Spatial Data Handling. IGU Commission on GIS*, pages 134–143, 1992.

[HS94] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.*, 4:63–98, 1994.

[HS95] John Hershberger and Subhash Suri. Morphing binary trees. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, pages 396–404, 1995.

[HU90] D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Intern. J. Computer Vision*, 5:195–212, 1990.

[II86] H. Imai and M. Iri. Computational-geometric methods for polygonal approximations of a curve. *Comput. Vision Graph. Image Process.*, 36:31–41, 1986.

[II88] H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. In G. T. Toussaint, editor, *Computational Morphology*, pages 71–86. North-Holland, Amsterdam, Netherlands, 1988.

[IR] S. Irani and P. Raghavan. Combinatorial and experimental results for randomized point matching algorithms. to appear.

[ISI89] K. Imai, S. Sumino, and H. Imai. Minimax geometric fitting of two corresponding sets of points. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 266–275, 1989.

[KCP92] J. Kent, W. Carlson, and R. Parent. Shape transformation for polyhedral objects. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 47–54, 1992.

[KR91] A. Kaul and J. Rossignac. Solid-interpolating deformations: construction and animation of PIPs. In *Proc. Eurographics*, pages 493–505, 1991.

[Lat91] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[LSW88a] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Object recognition by affine invariant matching. In *Proceedings of Computer Vision and Pattern Recognition*, pages 335–344, 1988.

[LSW88b] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. On recognition of 3-d objects from 2-d images. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 1407–1413, 1988.

[LW88] Y. Lamdan and H.J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision*, pages 238–249, 1988.

[Lyu66] L. A. Lyusternik. *Convex Figures and Polyhedra*. D. C. Heath, Boston, MA, 1966.

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

[MO88] A. Melkman and J. O'Rourke. On polygonal chain approximation. In G. T. Toussaint, editor, *Computational Morphology*, pages 87–95. North-Holland, Amsterdam, Netherlands, 1988.

[MS92] J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral surfaces. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms*, pages 296–306, 1992.

[NFWN94] R. Norel, D. Fischer, H. Wolfson, and R. Nussinov. Molecular surface recognition by a computer vision-based technique. *Protein Engineering*, 7:39–46, 1994.

[NLWN94] R. Norel, S.L. Lin, H. Wolfson, and R. Nussinov. Shape complimentarity at protein–protein interfaces. *Biopolymers*, 34:933–940, 1994.

[OY91] M. H. Overmars and C.-K. Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20:1034–1045, 1991.

[PY89] K. Przesławski and D. Yost. Continuity properties of selectors and Michael's theorem. *Michigan Math. J.*, 36:113–134, 1989.

[RK94] J. Rossignac and A. Kaul. Agrels and bips: Metamorphosis as a bézier curve in the space of polyhedra. In *Eurographics '94 Proceedings*, volume 13, pages 179–184, 1994.

[Sch88] S. Schirra. Über die Bitkomplexität der ε-Kongruenz. M.S. thesis, Univ. des Saarlandes, Computer Science Department, 1988.

[SG92] T. Sederberg and E. Greenwood. A physically based approach to 2D shape blending. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 25–34, 1992.

[SGWM93] T. Sederberg, P. Gao, G. Wang, and H. Mu. 2D shape blending: An intrinsic solution to the vertex path problem. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 15–18, 1993.

[She66] G. C. Shephard. The Steiner point of a convex polytope. *Canadian J. Math.*, 18:1294–1300, 1966.

[SR95] M. Shapira and A. Rappoport. Shape blending using the skeleton representation. *IEEE Computer Graphics and Appl.*, 16:44–50, 1995.

[Sur86] S. Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process.*, 35:99–110, 1986.

[SW94] J. Sprinzak and M. Werman. Affine point matching. *Pattern Recogn. Lett.*, 15:337–339, 1994.

[Var96] K. Varadarajan. Approximating monotone polygonal curves using the unfirm metric. In *Proc. 12th ACM Symp. Computational Geometry*, page to appear, 1996.

[WW93] Emo Welzl and Barbara Wolfers. Surface reconstruction between simple polygons via angle criteria. In *1st Annual European Symposium on Algorithms (ESA '93)*, volume 726 of *Lecture Notes in Computer Science*, pages 397–408. Springer-Verlag, 1993.