Symbolic and Numerical Computation for Artificial Intelligence

edited by

Bruce Randall Donald
Department of Computer Science
Cornell University, USA

Deepak Kapur
Department of Computer Science
State University of New York, USA

Joseph L. Mundy
AI Laboratory
GE Corporate R&D, Schenectady, USA

Academic Press
Harcourt Brace Jovanovich, Publishers
London San Diego New York
Boston Sydney Tokyo Toronto

ACADEMIC PRESS LIMITED
24–28 Oval Road
London NW1

US edition published by
ACADEMIC PRESS INC.
San Diego, CA 92101

This book is printed on acid-free paper

A catalogue record for this book is available from the British Library

ISBN 0-12-220535-9

Printed and Bound in Great Britain by
The University Press, Cambridge

Chapter 8

Distance Metrics for Comparing Shapes in the Plane

Daniel P. Huttenlocher[†]

Computer Science Department

Cornell University, Ithaca, NY 14853

Klara Kedem

Computer Science Department

Tel Aviv University, Tel Aviv, Israel

A central problem in pattern recognition, computer vision, and robotics is determining the extent to which one shape differs from another. We say that two geometric objects A and B have the same shape if there exists a transformation $T \in \mathcal{T}$ such that $T(A) = B$, where $\mathcal{T}$ is some transformation group (e.g. translation, similarity). That is, we define a shape to be an equivalence class of geometric objects under a given group of transformations. We are concerned with developing functions that measure the difference between two such shapes. We argue that for pattern matching applications it is important that such comparison functions be *metrics*. We show efficient methods of computing a number of different metrics on shapes, and present examples illustrating that these functions agree reasonably well with human intuition. In particular, we describe functions for comparing shapes composed of point sets in $\Re^d$ $(d = 2, 3)$ under translation, and for polygons under similarity transformations.

1. Introduction

Determining the degree of resemblance between two shapes is an important problem in a number of fields, including pattern recognition, computer vision and robotics. Recently we have been investigating *metrics* for comparing shapes (Arkin *et al.*, 1991; Huttenlocher and Kedem, 1990; Huttenlocher *et al.*, 1991a). Here we discuss some of these metrics and the methods for computing them. We define a shape to be an equivalence class of geometric objects, such that two objects A and B have the same shape exactly when there

[†] This work was supported in part by NSF grant IRI-9057928 and matching funds from General Electric and Kodak, and in part by the Air Force Office of Scientific Research under contract AFOSR-91-0328. The second author was supported by a fellowship from the Pikkowski-Valazzi Fund and by the Eshkol grant 04601-90.

exists some transformation $T \in \mathcal{T}$ such that $T(A) = B$, where $\mathcal{T}$ is a given transformation group. When A and B are the same shape according to this definition, we say $A \equiv B$. For example, the triangles A and B with side lengths $3, 4, 5$ and $6, 8, 10$ respectively have the same shape under a similarity transformation (translation, rotation and change of scale). In this paper we consider several types of geometric objects: (i) sets of points in $\Re^d$, $d = 2, 3$, (ii) sets of line segments in the plane, and (iii) simple polygons in the plane. We discuss methods for comparing shapes of the first two types under translation, and shapes of the third type under similarity transformation.

In pattern recognition and model-based vision applications, a stored set of 'model' shapes is often compared with an unknown shape that has been detected by some sensory device. The difference between each model shape and the unknown shape is computed, and the model that is closest to the unknown shape is reported as the best match. We have argued elsewhere (Arkin *et al.*, 1991) that for such applications the function used to measure the difference between shapes should be a metric (see Mumford, 1987) for similar arguments). This means that given a class of geometric objects the shape difference function d should obey the following properties, for any three shapes A, B and C,

1 $d(A, B) \geq 0$ for all A and B.
2 $d(A, B) = 0$ if and only if $A \equiv B$ (Identity).
3 $d(A, B) = d(B, A)$ for all A and B (Symmetry).
4 $d(A, B) + d(B, C) \geq d(A, C)$ for all A, B, and C (Triangle Inequality).

The triangle inequality is a particularly important property, because it guarantees that if several 'model' shapes are similar to a given instance then these shapes must also be similar to one another. Thus, for example, it is not possible for two highly dissimilar models to be both similar to the same instance. Current pattern recognition and model-based vision methods generally compare shapes using functions that are not metrics, and thus may report that several dissimilar models match the same input, which is highly counterintuitive. In addition to obeying metric properties, a shape comparison function should also be easy to compute in order for it to be of practical use. The functions that we describe can be computed efficiently both in theory and in practice.

We first discuss a metric for sets of points in $\Re^d$ under translation, and describe how to compute it efficiently for $d = 2, 3$. This distance is based on the Hausdorff metric, and was initially reported in Huttenlocher and Kedem (1990). For sets of points in the plane, the distance can be computed in time $O(pq(p+q)\log(pq))$ for the L_1, L_2, L_∞ metrics, where p and q are the number of points in the two sets being compared. For sets of points in three-space, the distance can be computed in time $O((pq)^2(p+q)\alpha(pq)\log^2(pq))$, where p and q are again the number of points in the two sets and where $\alpha(n)$ is the extremely slowly growing inverse Ackermann function. This technique for comparing point sets in the plane can be generalized to the problem of comparing sets of segments rather than sets of points; yielding an algorithm that runs in time $O((pq)^2\alpha(pq))$. These algorithms for computing the Hausdorff distance under translation are relatively involved to implement, and thus we also describe an approximation method that operates on a *rasterized* model and image (e.g. where the points can all be made to have integer coordinates). This is a provably good approximation scheme that runs very quickly in practice.

The second distance function that we discuss is for comparing polygonal shapes under

similarity transformations. This distance was originally reported in Arkin *et al.* (1991). The distance function can be computed in time $O(mn\log(mn))$ for two simple polygons with m and n vertices respectively. It is based on computing the L_2 distance between the turning function representations of the two polygons.

2. A Metric on Sets Under Translation

A shape comparison function $D(A, B)$ that measures the minimum Hausdorff distance between two point sets A and B under translation was defined in Huttenlocher and Kedem (1990). Here we present the definition of this function, claim that it is a metric, and describe methods for computing it efficiently.

The Hausdorff distance between two sets, $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$, where each a_i, b_j is either a point or a line segment, is given by

$$H(A, B) = \max(h(A^*, B^*), h(B^*, A^*)) \tag{2.1}$$

where A^* (resp. B^*) is the union of all points and segments in A (resp. B),

$$h(A^*, B^*) = \max_{a \in A^*} \min_{b \in B^*} \rho(a, b), \tag{2.2}$$

and $\rho(a, b)$ is the underlying metric. The function $h(A^*, B^*)$ is the *directed* Hausdorff distance from A^* to B^*, and measures the distance of the point of A^* that is farthest from any point of B^* (under ρ). Intuitively, the Hausdorff distance is small if and only if each point of A^* is near some point of B^* and vice versa (it measures the distance of the maximal outlying point).

It is well-known that the function $H(A, B)$ is a metric over the set of all closed, bounded sets. The Hausdorff distance, $H(A, B)$, can be trivially computed in time $O(pq)$ for two point sets of size p and q respectively; with some care, this can be improved to $O((p+q)\log(p+q))$ (Alt *et al.*, 1991).

The shape comparison function $D(A, B)$ is then defined to be the minimum value of the Hausdorff distance under translation. Without loss of generality we assume that the set A is fixed, and only the set B is allowed to translate, then

$$D(A, B) = \min_x H(A, B \oplus x) \tag{2.3}$$

where $B \oplus x = \{b + x | b \in B\}$, and H is the Hausdorff distance as defined above. That is, the distance is defined to be the minimal value of the Hausdorff distance over all possible translations of the set B.

CLAIM 2.1. *$D(A, B)$ is a metric.*

PROOF. Clearly $D(A, B)$ is everywhere non-negative, is symmetric and has the identity property because $H(A, B)$ is a metric and has these properties. We show that $D(A, B)$ also satisfies the triangle inequality.

Denote by x_{ac} the translation of A that minimizes $H(A, C)$, and similarly by x_{bc} the translation of B that minimizes $H(B, C)$. Let $A' = A \oplus x_{ab}$ and $B' = B \oplus x_{bc}$. Since the Hausdorff distance is a metric, the triangle inequality holds for H:

$$H(A', B') \leq H(A', C) + H(C, B'),$$

and by the choice of x_{ac} and x_{bc}

$$H(A', C) + H(C, B') = D(A, C) + D(C, B).$$

Since A' and B' are translations of A and B,

$$D(A, B) \leq H(A', B') \leq D(A, C) + D(C, B).$$

□

A problem closely related to measuring $D(A, B)$ is that of finding the best approximate congruence under translation for two sets of n points, A and B (Alt *et al.*, 1988). Formally, this problem is to find the translation x of B and the bijection $l : B \rightarrow A$ that minimizes

$$d = \max_{b \in B} \rho(b + x, l(b)).$$

This function is not very well suited to pattern recognition problems, however, because a sensing device will often merge two close points into one or split one point into two. When this happens there will either no longer be a total *matching* between the point sets, or the least cost matching will be forced to pair relatively distant points with one another and thus greatly increase the cost. In contrast, the minimum Hausdorff distance under translation just measures the proximity of each point in one set to the nearest point in the other — without requiring a matching between the sets.

2.1. The Minimum Hausdorff Distance for Point Sets

We now describe how to compute the minimum Hausdorff distance under translation, $D(A, B) = \min_x H(A, B \oplus x)$, for sets of points in $\Re^2$ and $\Re^3$. First we consider sets of points in the plane, and then show how the method generalizes to points in space. Note that because in this section the sets A and B contain only points, their unions A^* and B^* can be identified with A and B respectively. The main idea is to consider the distances defined in (2.1) and (2.2) as functions that depend on the translation x of the set B. These functions lead to constructs called *Voronoi surfaces*. Below we define Voronoi surfaces and the *upper envelope* (pointwise maximum) of a set of Voronoi surfaces. We then show the correspondence between the upper envelope of Voronoi surfaces and our problem of finding the minimum Hausdorff distance between sets of points.

Given a set $S = \{p_j | j = 1, \ldots, n\}$ of sources (points or line segments) in $\Re^d$, and some metric $\rho(.,.)$, the Voronoi *diagram* of S, denoted by $\mathrm{Vor}(S)$, is the decomposition of $\Re^d$ into 'Voronoi cells' $C_1, \ldots, C_n$, where each cell C_j contains those points of $\Re^d$ that are closer to p_j than to any other source (with closeness measured using the metric ρ). Consider now the function

$$d(x) = \min_{q \in S} \rho(x, q). \tag{2.4}$$

The graph of this function, $\{(x, d(x)) | x \in R^d\}$, is a surface which we call the *Voronoi surface* of S (we use a slight abuse of notation and refer to the surface also as $d(x)$). Note that this surface is at a local minimum (of zero) exactly when x is coincident with some source $p_j \in S$, and is at a local maximum for certain points that lie along the boundary of cells of $\mathrm{Vor}(S)$. That is, the surface gives the distance from x to the nearest point $p_j \in S$. An illustration of such a surface in one-dimension is shown in figure 1, where the four

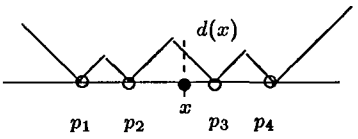

Figure 1. A Voronoi surface; the distance to the nearest point of S.

points $p_j \in S$ are denoted by open circles, and a particular point x is denoted by a closed circle. The height of the surface, $d(x)$, clearly gives the distance to the nearest point p_j. The analogous surface for sets of points in $\Re^2$ looks like an irregular 'egg-carton'.

The *upper envelope* (pointwise maximum) of m Voronoi surfaces was investigated in Huttenlocher *et al.* (1991a). This work provided bounds on the number of vertices of the upper envelope of m Voronoi surfaces, and presented theoretical algorithms for efficiently computing the upper envelope. We now discuss the relation between the problem of computing the minimum Hausdorff distance and the problem of computing the upper envelope of Voronoi surfaces. This yields algorithms for computing the minimum Hausdorff distance for sets of points in $\Re^2$ and $\Re^3$, and for sets of line segments in $\Re^2$.

In more detail, we can express the distance between a pair of points $a_i \in A$ and $b_j \in B$, as b_j undergoes a translation x, by

$$\delta_{i,j}(x) = \rho(a_i, b_j + x) = \rho(a_i - b_j, x)$$

where ρ is the underlying metric, which can be any L_p metric (but in this paper we will refer to the most common metrics, namely, L_1, L_∞ and L_2). We then define the function $d_i(x)$ to be the lower envelope of the functions $\delta_{i,j}(x)$ for a fixed point $a_i \in A$ and over all $b_j \in B$,

$$d_i(x) = \min_{b_j \in B} \delta_{i,j}(x). \tag{2.5}$$

If we denote the set $a_i \ominus B$ by S_i (i.e. $S_i = \{a_i - b_j | b_j \in B\}$) then substituting we obtain

$$d_i(x) = \min_{p \in S_i} \rho(p, x),$$

which is by definition the Voronoi surface of S_i from equation (2.4). Recall that this surface specifies the distance from a point x to the closest point of the set $S_i = a_i \ominus B$. Similarly, denoting the set $A \ominus b_j$ by S'_j, the function

$$d'_j(x) = \min_{a_i \in A} \delta_{i,j}(x) = \min_{p \in S'_j} \rho(p, x)$$

is the lower envelope of the functions $\delta_{i,j}(x)$ for a given $b_j \in B$ and over all $a_i \in A$.

Denote by $f(x)$ the upper envelope of the functions $d_i(x)$, $d'_j(x)$, then

$$f(x) = \max\left(\max_{a_i \in A} d_i(x), \max_{b_i \in B} d'_j(x)\right) = H(A, B \oplus x). \tag{2.6}$$

Hence

$$\min_x f(x) = \min_x H(A, B \oplus x).$$

Thus, in order to determine the minimum Hausdorff distance between two sets A and B, where the set B is translated by x, we have to identify the value of x that minimizes the

upper envelope of all the Voronoi surfaces defined by the sets $S_i = a_i \ominus B$ and $S'_j = A \ominus b_j$. Moreover, note that $f(x)$ specifies the value of $H(A, B \oplus x)$ for each translation x of the set B.

CLAIM 2.2. *The number of local minima of $f(x)$ is $O(pq(p+q))$ for the metrics L_1 and L_∞ as underlying metrics, and is $O(pq(p+q)\alpha(pq))$ for L_2 as the underlying metric.*

PROOF. It was shown in Huttenlocher *et al.* (1991a) that the upper envelope of m Voronoi surfaces with a total of n source points is of complexity $O(mn)$ for the L_1 and L_∞ metrics, and $O(mn\alpha(n))$ for the L_2 metric (where α is inverse Ackermann function). In computing the minimum Hausdorff distance we have $m = p+q$ sets (S_i, $1 \leq i \leq p$ and S_j, $1 \leq j \leq q$), and a total of $n = 2pq$ points over all the sets. Substituting these quantities, the result follows immediately. □

In order to determine $D(A, B)$ we must identify the global minimum of $f(x)$ which can be done by calculating all the local minima and inspecting each of them. Huttenlocher *et al.* (1991a) showed that the upper envelope of m Voronoi surfaces with a total of n source points can be computed in time $O(mn \log(n))$. Computing the upper envelope clearly dominates the running time, and thus,

CLAIM 2.3. *The minimum Hausdorff distance under translation between two sets of points in the plane (and the translation that achieves this minimum) can be computed in time $O(pq(p+q)\log(pq))$ for the metrics L_1, L_2, and L_∞.*

We do not present this algorithm here, however, because the method that we have implemented is an approximation based on rasterizing the Voronoi surfaces. The exact method requires the computation of $O(pq)$ Voronoi diagrams and the computation of $O(pq)$ unions of convex polygons having altogether $O(pq(p+q))$ edges and vertices, which in practice is not as fast as the rasterized method. Moreover, since most data from pattern matching and computer vision applications is already in raster form, the rasterized method is particularly appropriate.

When the sets $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$ consist of points in $\Re^3$, and the underlying distance metric ρ is L_2, we can apply the results of Huttenlocher *et al.* (1991a) on the upper envelope of four-dimensional Voronoi surfaces. They show that with m sets and a total of n source points, the complexity of computing the upper envelope in this case is $O(mn^2 \log(m)\alpha(n))$. For two sets of points A and B in space, with p and q points respectively, this yields the following bound (again $m = p+q$, $n = 2pq$),

CLAIM 2.4. *The minimum Hausdorff distance under translation between two sets of points in $\Re^3$ based on the L_2 metric (and the translation that achieves this minimum) can be computed in time $O((pq)^2(p+q)\alpha(pq)\log^2(p+q))$.*

2.2. THE MINIMUM HAUSDORFF DISTANCE FOR SETS OF SEGMENTS

The problem of computing the minimum Hausdorff distance for sets of line segments and points can also be solved by the technique of upper envelopes of Voronoi surfaces, although some extra care is needed here. The reason for the difficulties is that in the case

of line segments there is a substantial difference between the finite sets A, B and the infinite sets A^*, B^* (the unions of the points of A and of B respectively). Thus, direct application of the method in the previous section would call for computing the upper envelope of uncountably many Voronoi surfaces, each having a set of sources obtained as the Minkowski differences of segments in A and a point in B^*, or of a point in A^* and the segments of B. However, it suffices instead to form the upper envelope of only a finite number of surfaces, each obtained by computing the Minkowski sum of the Voronoi surface of the (reflected) set A with each segment of B, or the Minkowski sum of the Voronoi surface of (the reflected) set B with with each segment of A, as we describe below.

Let $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$ be two sets of points and segments in the plane, where we require that the segments are all open, and that if a set contains an (open) segment it also contains its endpoints; in other words, each closed line segment appears in a set as three distinct (and pairwise disjoint) sites—its relative interior and its endpoints (note that segments can share endpoints).

Define as above $A^* = \bigcup_{a \in A} a$, and $B^* = \bigcup_{b \in B} b$. We want to compute

$$D(A, B) = \min_x H(A, B \oplus x) =$$

$$\min_x \max(\max_{y \in A^*} \min_{z \in B^*} \rho(y, z + x), \max_{z \in B^*} \min_{y \in A^*} \rho(z + x, y))$$

where ρ is the underlying metric, which we assume in this subsection to be L_1 or L_∞ only. The above min-max-min expression is equal to

$$\min_x \max(\max_{a_i \in A} \max_{y \in a_i} \min_{z \in B^*} \rho(x, y - z), \max_{b_j \in B} \max_{z \in b_j} \min_{y \in A^*} \rho(x, y - z)).$$

Note that for any point $y \in a_i$ and for every x we have

$$\min_{z \in B^*} \rho(x, y - z) = \min_{b_j \in B} \rho(x, y - b_j) \ ,$$

and that the right hand side of this equation is the Voronoi surface of (the reflected) B translated by $y \in A^*$. Similarly, for the other minimization, we have for any point $z \in b_j$ and for every x, $\min_{y \in A^*} \rho(x, y - z) = \min_{a_i \in A} \rho(x, a_i - z)$, which is the Voronoi surface of $\mathrm{Vor}(A)$ translated by $z \in B^*$. That is, in the minimization portions of these expressions, we can minimize over (translated) objects in A or in B and not over their unions. We denote these Voronoi surfaces by

$$d_y(x) = \min_{b_j \in B} \rho(x, y - b_j) \qquad d_z(x) = \min_{a_i \in A} \rho(x, a_i - z)$$

for each $y \in A^*$, $z \in B^*$.

Next we define the upper envelope $f(x)$ of the surfaces $d_y(x)$ and $d_z(x)$ over all $y \in A^*$ and $z \in B^*$,

$$f(x) = \max(\max_{y \in A^*} d_y(x), \max_{z \in B^*} d_z(x)),$$

which we can rewrite as

$$f(x) = \max(\max_{a_i \in A} \max_{y \in a_i} d_y(x), \max_{b_j \in B} \max_{z \in b_j} d_z(x)).$$

It follows that $f(x)$ is the upper envelope of at most $p + q$ surfaces, each defined either

by $D_i(x) = \max_{y \in a_i} d_y(x)$, or by $D_j(x) = \max_{z \in b_j} d_z(x)$. $D_i(x)$ is the upper boundary of the volume obtained by sweeping the surface of $\mathrm{Vor}(-B)$ horizontally along a_i; $D_j(x)$ has a similar interpretation.

Let us fix a segment $a_i \in A$. Denote the endpoints of this segment by a' and a''. For each face F of $d_{a'}(x)$, when $y \in a_i$ moves from a' to a'', the face F is swept horizontally along the segment $a_i = a'' - a'$. The resulting swept volume is simply the Minkowski sum $F \oplus a_i$, which we denote by $\mathcal{F}$. For general metrics, including L_2, the structure of the swept volume is rather complicated, but in the L_1 and L_∞ metrics the structure is quite straightforward. In these cases each such face F is a polygon, hence the boundary of $\mathcal{F}$ is a prism whose two bases are parallel to F and all its other sides are parallelograms whose parallel edges are parallel to a_i. It follows that we can represent the swept surfaces as a collection of $O(pq)$ triangles: every D_i, for $i = 1, \ldots, p$ has $O(q)$ triangles, and every D_j for $j = 1, \ldots, q$, has $O(p)$ triangles, hence a total of $O(pq)$. This gets us to,

CLAIM 2.5. *The minimum Hausdorff distance under translation between two sets A and B, of p and q line segments respectively, with L_1 and L_∞ as underlying metrics, in the plane, can be computed in time $O((pq)^2 \alpha(pq))$.*

This is because the upper envelope $f(x)$ is the upper envelope of $O(pq)$ triangles, so its complexity is $O((pq)^2 \alpha(pq))$ (Pach and Sharir, 1989), and it can be computed in time $O((pq)^2 \alpha(pq))$ (Edelsbrunner *et al.*, 1989).

It is interesting to compare this result with recent results of Alt *et al.* (1991) for computing the minimum Hausdorff distance under translation, between sets of line segments in the plane, in time $O((pq)^3(p+q)\log(pq))$ for the L_2 metric (whereas Claim 2.5 is for L_1 and L_∞). Recently Agarwal *et al.* (1992) have come up with an algorithm that computes the minimum Hausdorff distance between sets of line segments in the plane under the L_2 metric in time $O((pq)^2(p+q)\log^3(pq))$, using parametric search.

2.3. COMPUTING THE HAUSDORFF DISTANCE IN PRACTICE

We now turn to the task of comparing two point sets A and B where the points of each set lie on an integer grid. For machine vision and pattern recognition applications this is a reasonable model, because most data comes from grid-based sensors. Thus we assume that we are given two sets of points $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$ such that each point $a_i \in A$ and $b_j \in B$ has integer coordinates. Consider the characteristic function of the set A,

$$a(x,y) = \begin{cases} 1 & \text{if } (x,y) \in A \\ 0 & \text{otherwise.} \end{cases}$$

This function is always zero for any non-integer values of x or y, because the set A is restricted to have points with integral coordinates. Thus the function can be represented using a binary array $A[k,l]$ where the k,l-th entry in the array is nonzero exactly when the point $(k,l) \in A$ (which of course is common in image processing applications). The set B has an analogously defined characteristic function and its array representation $B[s,t]$.

As in the continuous case considered above, we wish to compute the Hausdorff distance

as a function of translation by taking the pointwise maximum of a set of Voronoi surfaces for the point sets $S_i = \{a_i - b_j | b_j \in B\}$ and $S'_j = \{a_i - b_j | a_i \in A\}$. In this case, however, the point sets are now represented as binary arrays, where the nonzero elements of each array correspond to the elements of the sets. We denote by $S_i[x, y]$ the binary array representing the characteristic function of the set S_i, and by $S'_j[x, y]$ the characteristic function of the set S'_j. We then compute for each $S_i[x, y]$ the rasterized approximation to the Voronoi surface $d_i(x)$, which we denote by $D_i[x, y]$. This array specifies for each pixel location (x, y) the distance to the nearest nonzero point of $S_i[x, y]$ (for some distance function). That is, the array $D_i[x, y]$ is zero wherever $S_i[x, y]$ is one, and the other locations of $D_i[x, y]$ specify the distance to the nearest nonzero location in $S_i[x, y]$.

If we make the analogous definition of $D'_j[x, y]$ for the characteristic function $S'_j[x, y]$ of each set S'_j, then similarly to the previous section we can compute the pointwise maximum of these functions in order to determine the Hausdorff distance under translation (up to the rasterization accuracy of our integer grid),

$$F[x, y] = \max(\max_i D_i[x, y], \max_j D'_j[x, y]).$$

That is, $F[x, y]$ is an approximation to the Hausdorff distance as a function of translation, $f(x)$, given by equation (2.6). It can be shown that $|F[x_0, y_0] - f(x)| \leq 1.0$ for $x = (x_0, y_0)$ (Huttenlocher *et al.*, 1991b).

In the context of model-based recognition, we generally view the set B as being a 'model' that is matched under translation to an 'image' A (i.e. it is most natural to view the model as translating and the image as fixed). In order for $F[x, y]$, the Hausdorff distance, to be small at some translation (x_0, y_0), it must be that every $D_i[x, y]$ and $D'_j[x, y]$ are small at that location. This in turn means that every point of the translated model, B is near some point of A and vice versa.

Before turning to a discussion of the computation of $F[x, y]$, we briefly consider the computation of the digitized Voronoi surface, $D[x, y]$, using a distance transform (Borgefors, 1986), and using specialized hardware for computer graphics. The array $D[x, y]$, specifying the distance from each pixel (x, y) to the nearest nonzero pixel of a binary array $E[x, y]$, can be computed efficiently using an iterative local process. The value $D[x, y]$ is initially set to infinity if $E[x, y] = 1$ and one if $E[x, y] = 0$. Then the value of $D[x, y]$ at each pixel (x, y) is updated to be the minimum of its neighboring values, plus the distance from each neighbor to (x, y). This updating continues until no distance changes. Clearly at the end, each pixel reflects the distance to the nearest nonzero pixel of $E[x, y]$. The computation can also be done on a serial machine by making two passes over the distance array, as described in Borgefors (1986). These local propagation methods of computing $D[x, y]$ are only an approximation to the true distance when the L_2 norm (Euclidean metric) is used as the metric ρ specifying the distance between two points. When the L_1 norm or L_∞ norm are used to measure the distance between two points, then the computations yield the distance exactly.

The computation of $D[x, y]$ can also be performed very rapidly with special-purpose graphics hardware for doing z-buffer operations. The array $D[x, y]$ is simply the lower envelope of q cones (for the L_2 norm, or inverted pyramids for L_1 and L_∞), one for each point of B. Pointwise computation of upper and lower envelopes is exactly the operation that a z-buffer computes, when orthographic projection is used for the z-

buffering operation. Thus $D[x, y]$ can be computed rapidly by rendering q cones (or approximations thereof) and computing the view from $z = -\infty$.

2.3.1. Computing $F[x, y]$

In order to simplify the presentation, we consider separately the directed Hausdorff distance from A to B as a function of the translation of B, given by

$$F_A[x, y] = \max_i D_i[x, y],$$

and the directed Hausdorff distance from B to A given by $F_B[x, y] = \max_j D'_j[x, y]$ (where $D_i[x, y]$ and $D_j[x, y]$ are as defined above). Note that $F[x, y]$ is simply the pointwise maximum of these two directed distance functions.

The computation of $F_A[x, y]$ (resp. $F_B[x, y]$) simply involves computing the upper envelope of $D_i[x, y]$ shifted with respect to itself (resp. $D_j[x, y]$). This can be seen by noting that $S_i = \{a_i - b_j \mid b_j \in B\}$ (and $S'_j = \{a_i - b_j \mid a_i \in A\}$). Thus if we denote the distance transform (rasterized Voronoi surface) of $B[s, t]$ by $D[s, t]$, then $F_A[x, y]$ can be written equivalently as the maximization of the reflected $D[s, t]$ shifted by each nonzero value of $A[k, l]$,

$$F_A[x, y] = \max_i D_i[x, y] = \max_{k,l;A[k,l]=1} D[k - x, l - y] \tag{2.7}$$

(and similarly for $F_B[x, y]$). Note that this maximization can be performed very rapidly with special-purpose graphics hardware for doing pan (shift) and z-buffer operations.

It is also possible to view the computation of $F_A[x, y]$ slightly differently, and note that (2.7) is simply equivalent to maximizing the product of $A[k, l]$ and $D[s, t]$ at a given relative position,

$$F_A[x, y] = \max_k \max_l A[k, l] D[k - x, l - y]. \tag{2.8}$$

In other words, the maximization can be performed by 'positioning' the reflected $D[k, l]$ at each location (x, y), and computing the maximum of the product of A with D (and similarly for $F_B[x, y]$).

This form of the directed Hausdorff distance under translation is very similar to the binary correlation of the two arrays $A[k, l]$ and $B[s, t]$,

$$C[x, y] = \sum_k \sum_l A[k, l] B[k - x, l - y].$$

The only differences are that the array $B[s, t]$ in the correlation is replaced by the distance array $D[s, t]$ (the distance to the nearest pixel of $B[s, t]$), and the summation operations in the correlation are replaced by maximization operations. Binary correlation is one of the most commonly used tools in image processing, and thus it is interesting to briefly compare the Hausdorff distance under translation with correlation. One drawback of the correlation measure is that it is not a metric. In particular, this means that there may be several dissimilar models that all have high correlations with the same portion of the same image. Second, for binary images the correlation operation is quite sensitive to errors in the classification of a point (i.e. a 1-pixel that is classified as a 0 or vice versa), because it measures the exact superposition of points in $A[k, l]$ and $B[s, t]$. In contrast, the

minimum Hausdorff distance measures *nearby* points in the two sets, and thus is not very sensitive to pixel classification errors as long as nearby pixels were correctly classified. A final advantage of the (bidirectional) minimum Hausdorff distance over correlation is that it measures the degree of 'agreement' between the model and the image and vice versa. Thus it is possible to distinguish a situation in which a given model matches well from a situation in which literally any model will match well (such as a large region of all 'ones'). Correlation, which simply sums the product of the two bitmaps, cannot distinguish this.

Now we turn to a simple algorithm for computing the directed Hausdorff distance as a function of translation.

Algorithm 2.1. *Given two input binary image arrays, $A[k,l]$ and $B[s,t]$, compute the discrete directed Hausdorff distance under translation, $F_B[x,y]$, from $B[s,t]$ to $A[k,l]$ for each translation (x,y) of $B[s,t]$.*

1. Compute the array $D'[k,l]$ that specifies the distance to the closest nonzero pixel of $A[k,l]$ using a distance transform or special graphics hardware, as discussed above.
2. Denote the nonzero pixels of $B[s,t]$ by $(s_1,t_1),\ldots,(s_q,t_q)$. For each value (x,y)
   a. Compute the maximum over all values j, $1 \le j \le q$ of $B[s_j,t_j]D'[s_j+x,t_j+y]$.
   b. Set $F_B[x,y]$ to the maximum computed in the previous step.

The computation of $F_B[x,y]$ can be sped up substantially if there is a pre-specified threshold, ϵ, such that we are only interested in minima below this value. In such a case, the maximization in Step 2(a) only needs to be performed until some value over ϵ is seen (rather than for all values of j). More importantly, a large value of $F_B[x_0,y_0]$ means that neighboring locations around (x_0,y_0) can be ruled out. More specifically, if $F_B[x_0,y_0]=c$, then no points within radius $c-\epsilon$ can produce values of less than ϵ (because the slope of the function $F_B[x,y]$ is 1). In practice, it is often possible to determine such a threshold, ϵ, in which case the computation becomes extremely fast.

The bidirectional Hausdorff distance is the maximum of the directed Hausdorff distance from A to B and from B to A. However, in computing the bidirectional distance we must be sure to translate the sets A and B consistently. Simply using Algorithm 2.1as a subroutine ends up with inconsistent translations (the translations will have opposite signs from each other). The output $F[x,y]$ specifies a distance for each translation (x,y) of the model array $B[s,t]$.

Algorithm 2.2. *Given two input binary image arrays $A[k,l]$ and $B[s,t]$, compute the discrete Hausdorff distance under translation, $F[x,y]$, for each translation (x,y) of $B[s,t]$.*

1. Compute the array $D[s,t]$ that specifies the distance to the closest nonzero pixel of $B[s,t]$ using a distance transform or special graphics hardware, as discussed above.
2. Compute the analogous array $D'[k,l]$ for $A[k,l]$.
3. Denote the nonzero pixels of $A[k,l]$ by $(k_1,l_1),\ldots,(k_p,l_p)$, and denote the nonzero pixels of $B[s,t]$ by $(s_1,t_1),\ldots,(s_q,t_q)$. For each value (x,y)
   a. Compute the maximum over all values i, $1 \le i \le p$ of $A[k_i,l_i]D[k_i-x,l_i-y]$.
   b. Compute the maximum over all values j, $1 \le j \le q$ of $B[s_j,t_j]D'[s_j+x,t_j+y]$.
   c. Set $F[x,y]$ to the larger of the values computed in the previous two steps.

The bidirectional Hausdorff distance as computed by Algorithm 2.2 is not of much

practical use in many situations because there are often nonzero pixels of the image that have nothing to do with an instance of the object. In such cases, the distance "from the image to the model" will never be small, because these image pixels will not be near any nonzero pixels of the translated model. Thus, rather than maximizing over all nonzero pixels (k_i, l_i) of $A[k, l]$, it is more natural to maximize over those nonzero pixels of $A[k, l]$ that are covered by the translated array $B[s, t]$. This is analogous to the operation of correlation, where in effect a given translation of the model masks out all of the image except the portion that is covered by the model.

In many machine vision and pattern recognition applications it is also important to be able to identify instances of a model that are only partly visible (either due to occlusion or to failure of the sensing device to detect the entire object). The Hausdorff distance under translation can naturally be extended to the problem of finding the *best partial matches* between an 'model' bitmap $B[s, t]$ and a 'image' bitmap $A[k, l]$. Recall that for each location t the computation of $F[x, y]$ simply determines the distance of the point of the translated model $B \oplus t$ that is farthest from any point of the image A (and vice versa). Thus in effect each point of $B \oplus t$ is *ranked* by the distance to the nearest point of A (and vice versa). That is, the largest ranked point – the point farthest from any point of the other set – determines the distance. Hence, rather than maximizing over these rankings, it is possible to some compute quantile, or percentage value (for more details see Huttenlocher *et al.*, 1991b). This yields a natural notion of the best partial match at each translation.

We have implemented the above algorithms for computing the rasterized approximation to the Hausdorff distance under translation. In experiments contrasting the method with binary correlation, we have found that the Hausdorff distance is substantially less sensitive to small perturbations in the data than is correlation. The main reason is that the Hausdorff distance measures spatial proximity, whereas correlation measures exact superposition.

3. A Metric on Polygons Under Similarity Transformation

We now describe a metric for comparing polygonal shapes under similarity transformations (Arkin *et al.*, 1991). Consider a simple polygon represented as a sequence of points $P = (p_1, ..., p_n)$, $p_i \in \Re^2$. An alternative representation of P is the *turning function* $\Theta_P(s)$, which measures the *cumulative* angle of the counter-clockwise tangent as a function of the arc-length s (starting from some reference point on the boundary). That is, $\Theta_P(s)$ keeps track of the turning that takes place (see figure 2). Note that this is somewhat different from the standard definition, because we measure the cumulative angle rather than the angle between the tangent and the reference orientation. Without loss of generality, we assume that each polygon is rescaled so that the total perimeter length is 1; hence, Θ_P is a function from $[0, 1]$ to $\Re$.

We define the distance between two polygons P and Q to be the minimum of the L_2 distance between their two turning functions $\Theta_P(s)$ and $\Theta_Q(s)$, where the minimization is done over all possible relative orientations and all possible starting locations, s_0, along the boundary. Schwartz and Sharir (1984) have defined a similar distance function that is limited to comparing convex polygons. However, they compute an approximation based

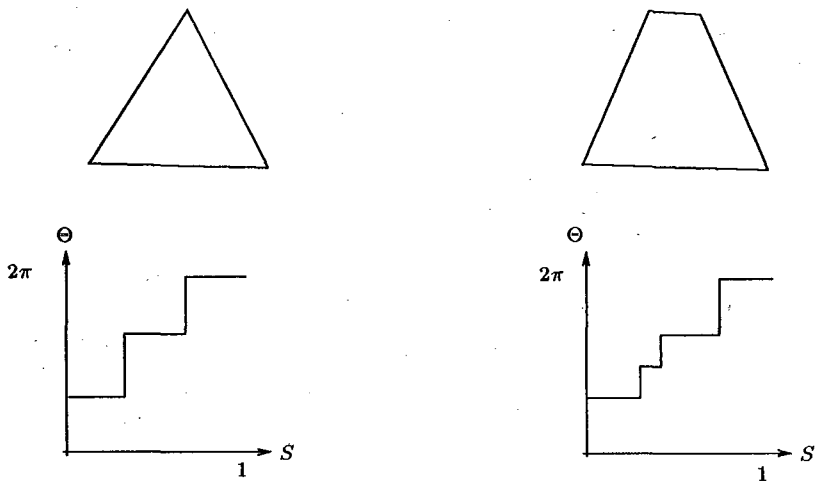

Figure 2. Two polygons and their turning functions, $\Theta(s)$.

on discretizing the turning functions into equally spaced points, where the quality of the approximation depends on the number of points chosen.

Consider two polygons P and Q and their associated turning functions $\Theta_P(s)$ and $\Theta_Q(s)$. The L_2 distance between $\Theta_P(s)$ and $\Theta_Q(s)$ is given by

$$\delta_2(P,Q) = \left(\int_0^1 |\Theta_P(s) - \Theta_Q(s)|^2 \, ds\right)^{\frac{1}{2}}$$

(Royden, 1968). δ_2 is by definition invariant with respect to translation and scaling of P and Q, but it is sensitive to both rotation and to choice of reference point on the boundary of either polygon. Since rotation and choice of reference point are arbitrary, it makes sense to consider the distance to be the minimum over all such choices. If we shift the reference point O along P's boundary by an amount t, then the new turning function is given by $\Theta_P(s+t)$. If we rotate P by an angle θ then the new function is given by $\Theta_P(s)+\theta$. Thus, we want to find the minimum over all such shifts t and rotations θ. In other words, we want to solve for

$$\begin{aligned} d_2(P,Q) &= \left(\min_{\theta\in\Re \; t\in[0,1]} \int_0^1 |\Theta_P(s+t) - \Theta_Q(s) + \theta|^2 \, ds\right)^{\frac{1}{2}} \\ &= \left(\min_{\theta\in\Re \; t\in[0,1]} h_{P,Q}(t,\theta)\right)^{\frac{1}{2}}, \end{aligned}$$

where

$$h_{P,Q}(t,\theta) = \int_0^1 |\Theta_P(s+t) - \Theta_Q(s) + \theta|^2 \, ds.$$

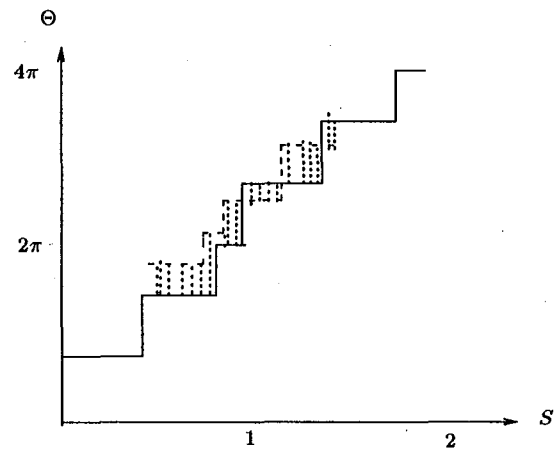

Figure 3. The rectangular strips formed by the functions $\Theta_P(s)$ (solid lines) and $\Theta_Q(s)$ (dashed lines). The shaded region between $\Theta_P(s+t)$ and $\Theta_Q(s)$ is the area being minimized.

CLAIM 3.1. *$d_2(P,Q)$ is a metric.*

PROOF. Clearly $d_2(.,.)$ is everywhere positive, is symmetric, and has the identity property, because the L_2 norm a metric and has these properties. By a straightforward application of the Minkowski inequality for L_p metrics, it can also be shown that $d_2(.,.)$ obeys the triangle inequality; see Arkin *et al.* (1991) for more detail. □

We can compute the value of $h_{P,Q}(t,\theta)$ for a fixed t simply by adding up the value of the integral within each *strip* defined by a consecutive pair of discontinuities in $\Theta_P(s)$ and $\Theta_Q(s)$ (see figure 3). The integral within a strip is trivially computed as the width of the strip times the square of the difference $|\Theta_P(s+t)-\Theta_Q(s)|$ (which is constant within each strip). Note that if m and n are the numbers of vertices in P and Q, respectively, then there are $m+n$ strips and that as θ changes, the value of the integral for each strip is a quadratic function of θ. Thus it is straightforward to show that,

CLAIM 3.2. *For any fixed value of t, $h_{P,Q}(t,\theta)$ is a quadratic function of θ.*

In order to compute $d_2(P,Q)$, we must minimize $h_{P,Q}(t,\theta)$ over all t and θ. We begin by finding the optimal θ for any fixed value of t. To simplify notation in the following discussion, we use $f(s)=\Theta_P(s)$, $g(s)=\Theta_Q(s)$, and $h(t,\theta)=h_{P,Q}(t,\theta)$.

CLAIM 3.3. *Let $h(t,\theta)=\int_0^1 (f(s+t)-g(s)+\theta)^2\,ds$. Then, in order to minimize $h(t,\theta)$, the best value of θ is given by*

$$\begin{aligned}\theta^*(t) &= \int_0^1 (g(s)-f(s+t))\,ds \\ &= \alpha - 2\pi t,\end{aligned}$$

where $\alpha=\int_0^1 g(s)\,ds-\int_0^1 f(s)\,ds$.

PROOF.

$$\begin{aligned}\frac{\partial h(t,\theta)}{\partial \theta} &= \int_0^1 (2\theta + 2f(s+t) - 2g(s))\, ds \\ &= 2\theta + 2\int_0^1 (f(s+t) - g(s))\, ds.\end{aligned}$$

Claim 3.2 assures us that the minimum occurs when we set this quantity equal to zero and solve for θ. Thus,

$$\theta^*(t) = \int_0^1 (g(s) - f(s+t))\, ds.$$

Some simplification of $\int_0^1 f(s+t)\, ds$ yields $2\pi t + \int_0^1 f(s)\, ds$. Thus,

$$\begin{aligned}\theta^*(t) &= \int_0^1 g(s)\, ds - 2\pi t - \int_0^1 f(s)\, ds \\ &= \alpha - 2\pi t.\end{aligned}$$

□

Substituting the expression for $\theta^*(t)$ in $d_2(P,Q)$ we are left with a one-variable minimization problem,

$$d_2(P,Q) = \left\{ \min_{t\in[0,1]} \left[\int_0^1 [f(s+t) - g(s)]^2\, ds - [\theta^*(t)]^2 \right] \right\}^{\frac{1}{2}}. \tag{3.1}$$

3.1. COMPUTING THE DISTANCE

In order to compute $d_2(P,Q)$ we show that the function we are minimizing, $h(t,\theta)$, achieves its minimum at one of mn discrete points on $[0,1]$, which we call *critical events*. Recall that in the process of finding $d_2(P,Q)$ we have to shift the function $f(s)$ to $f(s+t)$ for $t \in [0,1]$. During this shifting operation, the breakpoints of f collide with the breakpoints of g. We define a *critical event* as a value of t where a breakpoint of f collides with a breakpoint of g. Clearly there are mn such critical events for m breakpoints in f and n breakpoints in g. Using the fact that the minimum is obtained at a critical event, we show how to compute $d_2(P,Q)$ in time $O(n^2 \log(n))$ (or $O(mn \log(mn))$ for unequal numbers of vertices).

CLAIM 3.4. *If $f(\cdot)$ and $g(\cdot)$ are two piecewise-constant functions with m and n breakpoints respectively, then for constant θ,*

$$h(t,\theta) = \int_0^1 (f(s+t) - g(s) + \theta)^2\, ds$$

is piecewise-linear as a function of t, with mn breakpoints which are independent of the value θ.

PROOF. We give a geometric proof. First recall that for a given value of t the discontinuities in f and g define a set of $m+n$ rectangular strips (see figure 3). The value of $h(t,\theta)$ is simply the sum over all these strips of the width of a strip times the square of its height. Except at critical events, as f is shifted the width of each strip changes, but the height remains constant. Each changing rectangle contributes to changes in $h(t,\theta)$. If t is the amount of shift, then for a *shrinking* rectangle, the change is $(-t)$ times the square of the height; for a *growing* rectangle the change is $(+t)$ times the square of the height. Since the heights are constant, the change in $h(t,\theta)$ is a sum of linear terms and is therefore linear. Breakpoints in $h(t,\theta)$ clearly occur at each of the mn critical events where a discontinuity of f is aligned with a discontinuity of g. □

This result leads to a straightforward algorithm for computing $d_2(P,B)$. Let (t^*,θ^*) be the location of the minimum value of $h(t,\theta)$. By the preceding proposition, $h(t,\theta^*)$ is piecewise-linear as a function of t with breakpoints at a fixed set of critical values; thus, t^* must be at one of the critical values. Now, $h(t,\theta^*(t)) = h(t,0) - [\theta^*(t)]^2 = h(t,0) - [\alpha - 2\pi t]^2$ (from equation 3.1), so it suffices to evaluate $h(t,0) = \int [f(s+t) - g(s)]^2 ds$ at critical values of t. For each such value of t, recall that we can compute $h(t,0)$ in linear time, simply by adding up the squared heights of all of the strips. The optimal value $\theta^*(t)$ for each t can then be computed in constant time (by Claim 3.3). Thus the time for each critical event is linear, and the overall running time is $O(mn(m+n))$. This time bound can be improved by using a somewhat more complex algorithm.

CLAIM 3.5. *The distance $d_2(P,Q)$ between two polygons P and Q (with m and n vertices) can be computed exactly in time $O(mn\log(mn))$.*

PROOF. We show this by describing the algorithm. □

The basic idea is to compute $h(t,\theta^*(t))$ for each of the critical values of t. From the above discussion we know that it suffices to evaluate $h(t,0) = \int [f(s+t) - g(s)]^2 ds$ at critical values of t. Now we observe that by keeping track of a small set of values we can easily determine how the function $h(t,0)$ changes at each critical event. The values we keep track of are based on the rectangular strips that appear between the two functions $f(s)$ and $g(s)$. Recall that $g(s)$ is fixed in place and that $f(s)$ is shifted backwards by t. For a given value of t, the discontinuities in $f(s+t)$ and $g(s)$ define a set of rectangular strips, as was illustrated in figure 3. Each rectangular strip has f at the top and g at the bottom or vice-versa. The sides of a strip are determined by discontinuities in f and g.

We separate the strips into two groups based on the discontinuities at the sides of the strips: R_{fg} for those with f on the left and g on the right and R_{gf} for those with g on the left and f on the right. We keep track of two quantities: H_{fg} and H_{gf}. H_{fg} is the sum of the squares of the heights of all the strips in R_{fg}, and H_{gf} is the sum of the squares of the heights of all the strips in R_{gf}. The algorithm is based on the observation that for values of t between two critical events the slope of $h(t,0)$ is $H_{fg} - H_{gf}$. This follows from the fact that, as f is shifted backwards by t, R_{fg} is the set of all strips that increase in width by t, and R_{gf} is the set of all strips that decrease in width by t. The widths of the R_{ff} and R_{gg} strips remain unchanged.

Consider what happens at one of the critical events, where the change is no longer simply linear. We claim that the quantities H_{fg} and H_{gf} can be easily updated at these

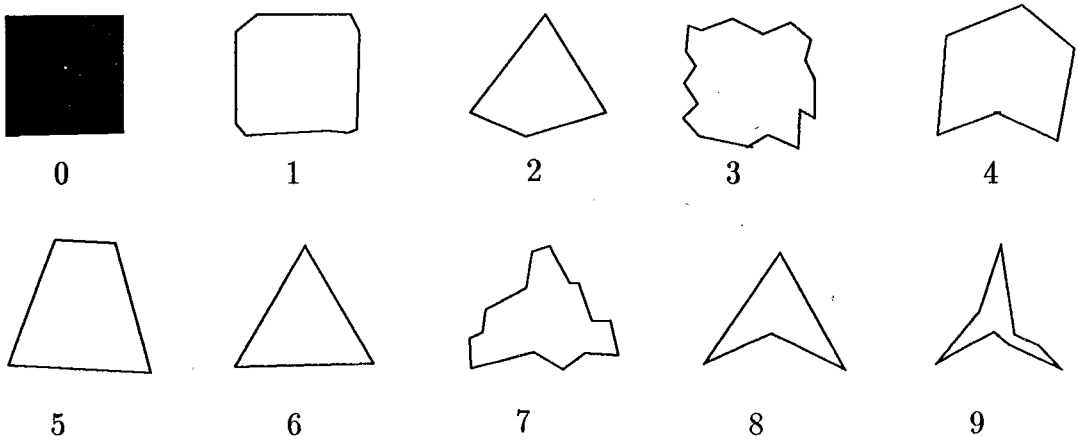

Figure 4. Comparing several polygons with a square using $d_2(P, Q)$.

points. To see this note that, at a critical event, a gf-type strip disappears (its width goes to zero) and a new fg-type strip appears (see figure 3). At the same time, the right boundary of the adjacent strip to the left is converted from g to f, and the left boundary of the adjacent strip to the right is converted from f to g. To update H_{fg} and H_{gf} we need to know just the values of f and g around the critical event.

Algorithm 3.1. *Given two polygons P and Q, compute the distance $d_2(P, Q)$.*

1. Compute the turning function representations, f and g, of the polygons P and Q, respectively.
2. Initialize:
   - Given the piecewise-constant functions f and g, determine the critical events: the shifts of f by t such that a discontinuity in f coincides with a discontinuity in g. Sort these critical events by how far f must be shifted for each event to occur. Let $c_0, c_1, \ldots, c_e$ be the ordered list of shifts for the critical events; $c_0 = 0$.
   - Calculate $h(0, 0)$. This involves summing the contributions of each of $m + n$ strips and takes linear time.
   - Determine initial values for H_{fg} and H_{gf}.
3. For $i = 1$ to e
   - Determine the value of
   $$h(c_i, 0) = (H_{fg} - H_{gf})(c_i - c_{i-1}) + h(c_{i-1}, 0).$$
   - Update H_{fg} and H_{gf}.

It is easy to see that the time for initialization is dominated by the time it takes to sort the critical events: $O(e \log e)$, where e is the number of critical events, or $O(mn \log(mn))$ where m and n are the sizes of the two polygons. The updates required for the remainder of the algorithm take a total of $O(e)$, or $O(mn)$ time.

We have implemented this method, and it runs quickly in practice, even for polygons with hundreds of vertices. We illustrate some of the qualitative aspects of the distance

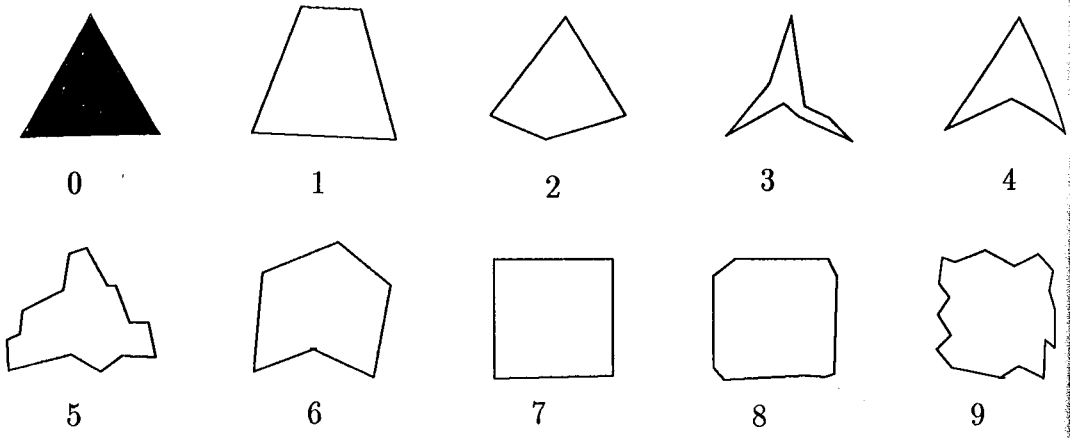

Figure 5. Comparing several polygons with a triangle using $d_2(P, Q)$.

function $d_2(P, Q)$ by comparing some simple polygons using the above algorithm. In addition to providing a distance, $d_2(P, Q)$, between two polygons, the method gives the relative orientation, θ^*, and the corresponding reference points of the two polygons for which this distance is attained. Consider the ten shapes shown in figures 4 and 5. In figure 4 the shapes are ordered by their distance from the square, and in figure 5 the same shapes are ordered by their distance from the triangle. (Note that the numbers under each shape reflect just the ordering, and not the magnitude of the distance.) The order of the shapes corresponds remarkably well to our intuitive idea of shape-resemblance. The match to the cut-off triangle suggests that the metric is useful for matching partially occluded objects, as long as the overall shape of the object does not change too radically.

A straightforward extension of the algorithm applies to shapes composed of piecewise circular arcs rather than line segments. In this case, the function $\Theta_P(s)$ is piecewise linear rather than piecewise constant.

4. Summary

We have discussed a number of methods for comparing shapes. We defined two geometric objects A and B to have the same shape, $A \equiv B$, if they are in the same equivalence class with respect to a given transformation group (i.e. if $T(A) = B$ for some transformation T in the group). We have argued that shape comparison functions should obey metric properties, and have presented several distance metrics that are efficiently computable both in theory and practice. The methods are applicable to problems in pattern recognition, computer vision, and robotics. In particular, we described a function for comparing sets of points or line segments in the plane using the Hausdorff distance as a function of translation. This shape comparison function can be computed efficiently in theory, and a close approximation can be computed efficiently in practice. We also investigated how this method can be extended to sets of points in $\Re^3$. The second class of shape comparison functions we discussed are based on comparing the turning function

representations of polygons. This latter class of methods does not appear to extend easily to shapes in higher dimensions, which is an important area for many applications.

References

P.K. Agarwal, M. Sharir and S. Toledo (1992), "Applications of parametric searching in geometric optimization", *Proc. 3rd ACM-SIAM Symp. Discrete Algorithms*, to appear.

H. Alt, B. Behrends and J. Blomer (1991), "Measuring the resemblance of polygonal shapes", *Proc. 7th ACM Symp. Computational Geom.*, N. Conway, NH, 186–193.

H. Alt, K. Mehlhorn, H. Wagener and E. Welzl (1988), "Congruence, similarity, and symmetries of geometric objects", *Discrete and Computational Geom.*, **3**, 237–256.

E. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem and J.S.B Mitchell (1991),"An efficiently computable metric for comparing polygonal shapes", *IEEE Trans. Patt. Anal. Mach. Intell.*, **13**(3), 209–216.

G. Borgefors (1986), "Distance transforms in digital images", *Comput. Vision Graph. Image Processing*, **34**, 344–371.

H. Edelsbrunner, L.J. Guibas and M. Sharir (1989), "The upper envelope of piecewise linear functions", *Discrete and Computational Geom.*, **4**, 311–336.

H. Edelsbrunner and R. Seidel (1986), "Voronoi diagrams and arrangements", *Discrete and Computational Geom.*, **1**, 25–44.

D.P. Huttenlocher and K. Kedem (1990), "Computing the minimum hausdorff distance for point sets under translation", *Proc. 6th ACM Symp. Computational Geom.*, Berkeley, CA, 340–349.

D.P. Huttenlocher, K. Kedem and M. Sharir (1991a), "The upper envelope of Voronoi surfaces and its applications", *Proc. 7th ACM Symp. Computational Geom.*, N. Conway, NH, 194–293.

D.P. Huttenlocher, G.A. Klanderman and W.J. Rucklidge (1991b), *Comparing Images Using the Hausdorff Distance Under Translation*, Technical Report 91-1211, Dept. of Comput. Sci., Cornell University, Ithaca, NY.

D. Mumford (1987), "The problem of robust shape descriptors", *Proc. 1st Int. Conf. Comput. Vision*, IEEE Comput. Soc. Press, 602–606.

J. Pach and M. Sharir (1989), "The upper envelope of piecewise linear functions and the region enclosed by convex plates, I: combinatorial analysis", *Discrete and Computational Geom.*, **4**, 291–309.

H.L. Royden (1968), *Real Analysis*, Macmillan, NY.

J.T. Schwartz and M. Sharir (1984), *Some Remarks on Robot Vision*, Technical Report 119, Robotics Report 25, Courant Institute of Math. Sci., New York University, NY.