Comparing Images Using the Hausdorff Distance[*]

Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge

Department of Computer Science
Cornell University
Ithaca, NY 14853

CUCS TR 91-1211 (revised)

Abstract

The Hausdorff distance measures the extent to which each point of a 'model' set lies near some point of an 'image' set and vice versa. Thus this distance can be used to determine the degree of resemblance between two objects that are superimposed on one another. In this paper we provide efficient algorithms for computing the Hausdorff distance between all possible relative positions of a binary image and a model. We focus primarily on the case in which the model is only allowed to translate with respect to the image. Then we consider how to extend the techniques to rigid motion (translation and rotation). The Hausdorff distance computation differs from many other shape comparison methods in that no correspondence between the model and the image is derived. The method is quite tolerant of small position errors as occur with edge detectors and other feature extraction methods. Moreover, we show how the method extends naturally to the problem of comparing *a portion* of a model against an image.

[*]This work was supported in part by NSF grant IRI-9057928 and matching funds from General Electric, Kodak and Xerox, and in part by contract AFOSR-91-0328. WJR was also supported by a Natural Sciences and Engineering Research Council (Canada) 1967 Scholarship

1 Introduction

A central problem in pattern recognition and computer vision is determining the extent to which one shape differs from another. Pattern recognition operations such as correlation and template matching (cf. [17]) and model-based vision methods (cf. [4, 8, 11]) can all be viewed as techniques for determining the difference between shapes. We have recently been investigating functions for determining the degree to which two shapes differ from one another. The goal of these investigations has been to develop shape comparison methods that are efficient to compute, produce intuitively reasonable results, and have a firm underlying theoretical basis. In order to meet these goals, we argue that it is important for shape comparison functions to obey metric properties (see [3] for related arguments).

In this paper we present algorithms for efficiently computing the Hausdorff distance between all possible relative positions of a model and an image. (The Hausdorff distance is a max-min distance defined below.) We primarily focus on the case in which the model and image are allowed to translate with respect to one another, and then briefly consider extensions to handle the more general case of rigid motion. There are theoretical algorithms for efficiently computing the Hausdorff distance as a function of translation [12, 14] and rigid motion [13]. Here we provide provably good approximation algorithms that are highly efficient both in theory and in practice. These methods operate on binary rasters, making them particularly well suited to image processing and machine vision applications, where the data are generally in raster form. The three key advantages of the approach are: (i) relative insensitivity to small perturbations of the image, (ii) simplicity and speed of computation, and (iii) naturally allowing for *portions* of one shape to be compared with another.

We discuss three different methods of computing the Hausdorff distance as a function of the translation of a model with respect to an image. The first of these methods is similar in many ways to binary correlation and convolution, except that the Hausdorff distance is a *nonlinear* operator. The second method extends the definition of the distance function to enable the comparison of portions of a model to portions of an image. The third method improves on the first two, by using certain properties of the Hausdorff distance to rule out many possible relative positions of the model and the image without having to explicitly consider them. This speeds up the computation by several orders of magnitude. All three of these methods can be further sped up using special-purpose graphics hardware (in particular a z-buffer). We present a number of examples using real images. These examples illustrate the application of the method to scenes in which a portion of the object to be identified is hidden from view. Then finally we show how the methods can be adapted to comparing objects under rigid motion (translation and rotation).

1.1 The Hausdorff Distance

Given two finite point sets $A = \{a_1, \dots, a_p\}$ and $B = \{b_1, \dots, b_q\}$, the Hausdorff distance is defined as

$$H(A, B) = \max(h(A, B), h(B, A)) \tag{1}$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|, \tag{2}$$

and $\| \cdot \|$ is some underlying norm on the points of A and B (e.g., the L_2 or Euclidean norm).

The function $h(A, B)$ is called the *directed* Hausdorff distance from A to B. It identifies the point $a \in A$ that is farthest from any point of B, and measures the distance from a to its nearest neighbor in B (using the given norm $\| \cdot \|$). That is, $h(A, B)$ in effect ranks each point of A based on its distance to the nearest point of B, and then uses the largest ranked such point as the distance (the most mismatched point of A). Intuitively, if $h(A, B) = d$, then each point of A must be within distance d of some point of B, and there also is some point of A that is exactly distance d from the nearest point of B (the most mismatched point).

The Hausdorff distance, $H(A, B)$, is the maximum of $h(A, B)$ and $h(B, A)$. Thus it measures the degree of mismatch between two sets, by measuring the distance of the point of A that is farthest from any point of B and vice versa. Intuitively, if the Hausdorff distance is d, then every point of A must be within a distance d of some point of B and vice versa. Thus the notion of resemblance encoded by this distance is that each member of A be near some member of B and vice versa. Unlike most methods of comparing shapes, there is no explicit pairing of points of A with points of B (for example many points of A may be close to the *same* point of B). The function $H(A, B)$ can be trivially computed in time $O(pq)$ for two point sets of size p and q respectively, and this can be improved to $O((p + q) \log(p + q))$ [2].

It is well known that the Hausdorff distance, $H(A, B)$, is a metric over the set of all closed, bounded sets (cf., [9]). Here we restrict ourselves to finite point sets, because that is all that is necessary for raster sensing devices. It should be noted that the

Hausdorff distance does not allow for comparing portions of the sets A and B, because every point of one set is required to be near some point of the other set. There is, however, a natural extension to the problem of measuring the distance between some subset of the points in A and some subset of the points in B, which we present in Section 3.

The Hausdorff distance measures the mismatch between two sets that are at *fixed* positions with respect to one another. In this paper we are primarily interested in measuring the mismatch between *all possible* relative positions of two sets, as given by the value of the Hausdorff distance as a function of relative position. That is, for any group G, we define the minimum Hausdorff distance to be

$$M_G(A, B) = \min_{g_1, g_2 \in G} H(g_1 A, g_2 B).$$

If the group G is such that, for any $g \in G$ and for any points x_1, x_2, $\|gx_1 - gx_2\| = \|x_1 - x_2\|$, then we need only consider transforming one of the sets:

$$M_G(A, B) = \min_{g \in G} H(A, gB).$$

This property holds when G is the group of translations and $\|\cdot\|$ is any norm, and also when G is the group of rigid motions and $\|\cdot\|$ is the Euclidean norm.

In the cases we consider here (translations and rigid motions), the minimum Hausdorff distance obeys metric properties (as was shown in [12] and [13]). That is, the function is everywhere positive, and has the properties of identity, symmetry and triangle inequality. These properties correspond to our intuitive notions of shape resemblance, namely that a shape is identical only to itself, the order of comparison of two shapes does not matter[1] and two shapes that are highly dissimilar cannot both be similar to some third shape. This final property, the triangle inequality, is particularly important in pattern matching applications where several stored model shapes are compared to an unknown shape. Most shape comparison functions used in such applications do not obey the triangle inequality, and thus can report that two highly dissimilar model shapes are both similar to the unknown shape. This behavior is highly counter-intuitive (for example, reporting that some unknown shape closely resembles both an 'elephant' and a 'hatrack' is not desirable, because these two shapes are highly dissimilar).

[1]Actually the order of comparison does matter in some psychophysical studies. One interesting property of the Hausdorff distance in this regard is the fact that the *directed* distance $h(A, B)$ is not symmetric.

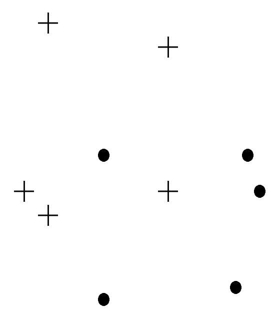

Figure 1: Two sets of points illustrating the distances $H(A, B)$ and $M_T(A, B)$.

We focus primarily on the case where the relative positions of the model with respect to the image is the group of translations. Without loss of generality we fix the set A and allow only B to translate. The minimum value of the Hausdorff distance under translation is then defined as,

$$M_T(A, B) = \min_t H(A, B \oplus t) \tag{3}$$

where H is the Hausdorff distance as defined in equation (1), and $\oplus$ is the standard Minkowski sum notation (i.e., $B \oplus t = \{b + t \mid b \in B\}$). For example, Figure 1 shows two sets of points, where the set A is illustrated by dots and the set B by crosses. $H(A, B)$ is large because there are points of A that are not near any points of B and vice versa. $M_T(A, B)$ is small, however, because there is a translation of B that makes each point of A nearly coincident with some point of B and vice versa. We generally refer to the set A as the 'image' and the set B as the 'model', because it is most natural to view the model as translating with respect to the image.

We now turn to the problem of efficiently computing $H(A, B)$ and $M_T(A, B)$. The organization of the remainder of the paper is as follows. We first discuss how to compute $H(A, B)$ for finite sets of points in the plane. The basic idea is to define a set of functions measuring the distance from each point of A to the closest point of B (and vice versa), as a function of the translation t of the set B. In section 3 we show how to extend the method to the problem of comparing *portions* of the sets A and B (e.g., as occurs when instances are partly occluded). Then in section 4 we discuss an implementation of the Hausdorff distance computation for raster data, where the sets A and B are represented in terms of binary rasters. This implementation is in many ways similar to binary correlation. In section 5 we show how to improve the basic implementation, by ruling out many possible translations of B without explicitly considering them. We then present some examples, and contrast the method with binary correlation. Finally, we consider the case of rigid motion (translation and rotation), and present an example for this problem.

2 Computing $H(A, B)$ and $M_T(A, B)$

From the definition of the Hausdorff distance in equations (1) and (2), we have

$$\begin{aligned} H(A,B) &= \max(h(A,B), h(B,A)) \\ &= \max\left(\max_{a\in A}\min_{b\in B}\|a-b\|, \max_{b\in B}\min_{a\in A}\|a-b\|\right) \end{aligned}$$

If we define $d(x) = \min_{b\in B}\|x-b\|$ and $d'(x) = \min_{a\in A}\|a-x\|$, we have

$$H(A,B) = \max\left(\max_{a\in A} d(a), \max_{b\in B} d'(b)\right).$$

That is, $H(A, B)$ can be obtained by computing $d(a)$ and $d'(b)$ for all $a \in A$ and $b \in B$ respectively. The graph of $d(x)$, $\{(x, d(x)) \mid x \in \Re^2\}$, is a surface that has been called the *Voronoi surface* of B [14]. This surface gives for each location x the distance from x to the nearest point $b \in B$. For points in the plane one can visualize this surface as a sort of 'egg carton', with a local minimum of height zero corresponding to each $b \in B$, and with a 'cone-shape' rising up from each such minimum. The locations at which these cone-shapes intersect define the local maxima of the surface. Thus note that the local maxima are equidistant from two or more local minima (hence the name Voronoi surface, by analogy to Voronoi diagrams that specify the locations equidistant from two or more points of a given set [16]). The graph of $d'(x)$ has a similar shape, with a 'cone-shape' rising up from each point of A.

A Voronoi surface, $d(x)$, of a set B has also been referred to as a *distance transform* (e.g., [10]), because it gives the distance from any point x to the nearest point in a set of source points, B. Figure 2 illustrates a set of points and a top-down view of a corresponding Voronoi surface, where brighter (whiter) portions of the image correspond to higher portions of the surface. The norm used in the figure is L_2.

We now turn to calculating the Hausdorff distance as a function of translation,

$$\begin{aligned} H(A, B\oplus t) &= \max\left(\max_{a\in A}\min_{b\in B}\|a-(b+t)\|, \max_{b\in B}\min_{a\in A}\|a-(b+t)\|\right) \\ &= \max\left(\max_{a\in A}\min_{b\in B}\|(a-t)-b\|, \max_{b\in B}\min_{a\in A}\|a-(b+t)\|\right) \\ &= \max\left(\max_{a\in A} d(a-t), \max_{b\in B} d'(b+t)\right) \end{aligned}$$

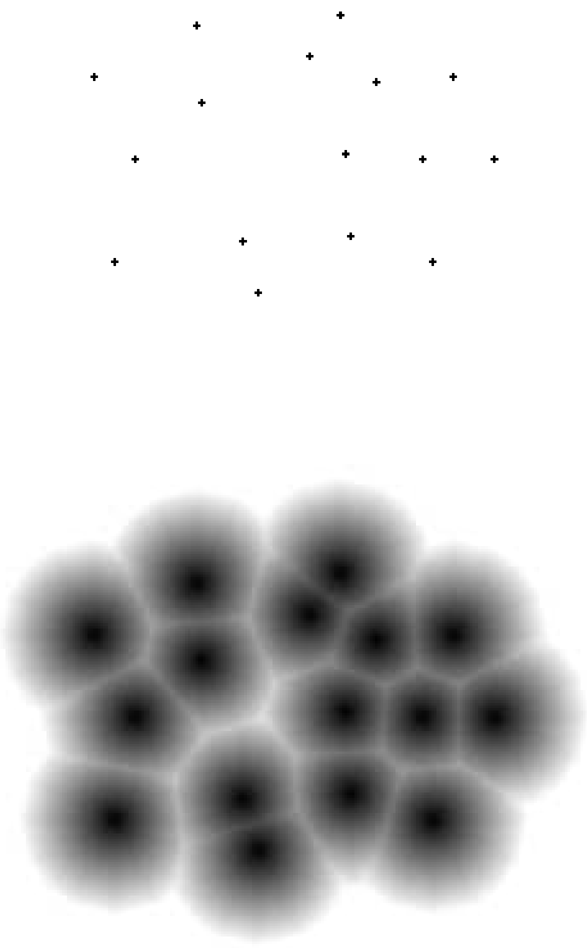

Figure 2: A set of points and a corresponding Voronoi surface.

That is, $H(A, B \oplus t)$ is simply the maximum of translated copies of the Voronoi surfaces $d(x)$ and $d'(x)$ (of the sets B and A respectively). Now define $f_A(t) = \max_{a \in A} d(a - t)$, the upper envelope (pointwise maximum) of p copies of the function $d(-t)$, which have been translated relative to each other by each $a \in A$. This gives for each translation t the distance of the point of A that is farthest from any point of $B \oplus t$. That is, $f_A(t) = h(A, B \oplus t)$, where $h(\cdot, \cdot)$ is the directed Hausdorff distance given in equation (2).

The directed Hausdorff distance $f_B(t) = h(B \oplus t, A)$ is defined analogously. Now $H(A, B \oplus t)$ is simply the maximum of the two directed distance functions. Thus, we define

$$\begin{aligned} f(t) &= \max(f_A(t), f_B(t)) \\ &= \max\left(\max_{a \in A} d(a-t),\ \max_{b \in B} d'(b+t)\right) \\ &= H(A, B \oplus t), \end{aligned}$$

where $H(\cdot, \cdot)$ is the Hausdorff distance as defined in (1). That is, the function $f(t)$ specifies the Hausdorff distance between two sets A and B as a function of the translation t of the set B. Efficient theoretical algorithms for computing $f(t)$ were developed in [14]. There it was shown that if A and B contain respectively p and q points in the plane, then the function $f(t)$ can be computed in time $O(pq(p+q) \log pq)$ when $\| \cdot \|$ is the L_1, L_2 or L_∞ norm. This running time can be improved to $O(pq \log pq)$ when using the L_1 or L_∞ norms [7]. These methods are quite complicated to implement, and are substantially less efficient in practice than the rasterized approximation methods that we investigate in sections 4 and 5.

3 Comparing Portions of Shapes

In many machine vision and pattern recognition applications it is 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). Thus we wish to extend the definition of the Hausdorff distance to allow for the comparison of *portions* of two shapes. This will allow both for scenes that contain multiple objects, and for objects that are partially hidden from view.

3.1 Partial distances based on ranking

The Hausdorff distance can naturally be extended to the problem of finding the *best partial distance* between a model set B and an image set A. For simplicity we first consider just the directed Hausdorff distance from B to A, $h(B, A)$. The computation of $h(B, A)$ simply determines the distance of the point of the model B that is farthest from any point of the image A. That is, each point of B is *ranked* by the distance to the nearest point of A, and the largest ranked point (the one farthest from any point of A) determines the distance.

Thus a natural definition of 'distance' for K of the q model points $(1 \leq K \leq q)$ is given by taking the K-th ranked point of B (rather than the largest ranked one),

$$h_K(B, A) = \underset{b \in B}{K^{\text{th}}} \min_{a \in A} \|a - b\|, \tag{4}$$

where ${K^{\text{th}}}_{b \in B}$ denotes the K-th ranked value in the set of distances (one corresponding to each element of B). That is, for each point of B the distance to the closest point of A is computed, and then the points of B are ranked by their respective values of this distance. The K-th ranked such value, d, tells us that K of the model points B are each within a distance d of some image point (and when $K = q$ all the points are considered, and the value is simply the directed Hausdorff distance $h(B, A)$). This definition of the distance has the nice property that it *automatically selects* the K 'best matching' points of B, because it identifies the subset of the model of size K that minimizes the directed Hausdorff distance.

In general, in order to compute the partial directed 'distance' $h_K(B, A)$, we specify some fraction $0 \leq f_1 \leq 1$ of the points of B that are to be considered. Each of the q points of B is ranked by the distance to the nearest point of A. The K-th ranked such value, given by equation (4), then gives the partial 'distance', where $K = \lfloor f_1 q \rfloor$. This K-th ranked value can be computed in $O(q)$ time using standard methods such as those in [1]. In practice, it takes about twice as long as computing the maximum.

This partial distance measures the difference between a *portion* of the model and the image: the K points of the model set which are closest to points of the image set. One key property of this method is that it does not require one to pre-specify which part of the model is to be compared with the image. This is because the computation of the directed Hausdorff distance determines how far each model point is from the nearest image point, and thus automatically selects the K points of the model that are closest to image points. In Section 6 we illustrate this partial matching capability, and contrast it with correlation. We find that the directed partial Hausdorff 'distance' works well for partial matches on images where correlation does not.

The partial *bidirectional* Hausdorff 'distance' is now naturally defined as

$$H_{LK}(A,B) = \max(h_L(A,B), h_K(B,A)). \tag{5}$$

This function clearly does not obey metric properties, however it does obey weaker conditions that provide for intuitively reasonable behavior. These conditions are, in effect, that metric properties are obeyed between given subsets of A and B (of size L and K respectively). In order to specify these properties more precisely, we need to understand something about the subsets of A and B that achieve the minimum partial 'distance':

Claim 1 *If $H_{LK}(A,B) = d$ then there exist sets $A_L \subseteq A$ and $B_K \subseteq B$ such that $H(A_L, B_K) \leq d$. Each of A_L and B_K will have exactly $\min(K,L)$ elements.*

There are also "large" subsets of A and B which achieve the distance d:

Claim 2 *If $H_{LK}(A,B) = d$ then there exist sets $A'_L \subseteq A$ and $B'_K \subseteq B$ such that $H(A'_L, B'_K) = d$, with $L \leq |A'_L| \leq \max(K,L)$ and $K \leq |B'_K| \leq \max(K,L)$.*

For proofs of these claims, see Appendices A and B.

It follows immediately that the identity and symmetry properties hold with respect to A'_L and B'_K, because $H(\cdot,\cdot)$ obeys these properties. Intuitively, this means that for the partial 'distance' with some given K, L, the order of comparison does not matter, and the distance is zero exactly when the two minimizing subsets A'_L and B'_K are the same.

For the triangle inequality the minimizing subsets may be different when comparing A with B than when comparing B with C. Thus in general the triangle inequality can be violated. In the restricted case that the same subset $B'_K \subseteq B$ is the minimizer of $H_{LK}(A,B)$ and $H_{KM}(B,C)$ then by definition $H(A'_L, B'_K) + H(B'_K, C'_M) \geq H(A'_L, C'_M)$. Intuitively, this means that if two sets are compared with the *same portion* of a third set (denoted B'_K above) then the triangle inequality holds. For practical purposes this is a reasonable definition: if two models both match the same part of a given image then we expect the models to be similar to one another. On the other hand if they match different parts of an image then we have no such expectation.

4 The Minimum Hausdorff Distance for Grid Points

We now turn to the case in which the point sets lie on an integer grid. This is appropriate for many computer vision and pattern recognition applications, because the data are derived from a raster device such as a digitized video signal. Assume we are given two sets of points $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$ such that each point $a \in A$ and $b \in B$ has integer coordinates. We will denote the Cartesian coordinates of a point $a \in A$ by (a_x, a_y), and analogously (b_x, b_y) for $b \in B$. The characteristic function of the set A 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 $(k, l) \in A$ (as is standard practice). The set B has an analogously defined array representation $B[k, l]$.

As in the continuous case in the previous section, we wish to compute the Hausdorff distance as a function of translation by taking the pointwise maximum of a set of Voronoi surfaces. In this case, however, the point sets from which these surfaces are derived are represented as arrays, where the nonzero elements of the arrays correspond to the elements of the sets.

For the two sets A and B, we compute the rasterized approximations to their respective Voronoi surfaces $d'(x)$ and $d(x)$. These distance arrays, or distance transforms, specify for each pixel location (x, y) the distance to the nearest nonzero pixel of A or B respectively. We use the notation $D'[x, y]$ to denote the distance transform of $A[k, l]$, and $D[x, y]$ to denote the distance transform of $B[k, l]$. That is, the array $D'[x, y]$ is zero wherever $A[k, l]$ is one, and the other locations of $D'[x, y]$ specify the distance to the nearest nonzero point of $A[k, l]$. There are a number of methods for computing the rasterized Voronoi surface, or distance transform (e.g., [5, 15]), which we discuss briefly below.

Proceeding with the analogy to the continuous case, we can compute the pointwise maximum of all the translated D and D' arrays to determine the Hausdorff distance as a function of translation (only now we are limited by the rasterization accuracy of the integer grid):

$$F[x, y] = \max\left(\max_{a \in A} D[a_x - x, a_y - y],\ \max_{b \in B} D'[b_x + x, b_y + y]\right). \tag{6}$$

In order for $F[x, y]$ to be small at some translation (x_0, y_0), it must be that the distance transform $D[x, y]$ is small at all the locations $A \ominus (x_0, y_0)$ and that $D'[x, y]$ is small at all the locations $B \oplus (x_0, y_0)$, In other words, every point (nonzero pixel) of the translated model array $B[k + x_0, l + y_0]$ must be near some point (nonzero pixel) of the image array $A[k, l]$, and vice versa.

When the input points have integer coordinates, it is straightforward to show that the minimum value of $F[x, y]$ is very close to the minimum value of the exact function $f(t)$. In other words, the rasterization only introduces a small error compared to the true distance function. Specifically,

Claim 3 *Let $t_0 = (x_0, y_0)$ be a translation which minimizes $F[x, y]$. (There may be more than one translation with the same, minimum, F value). Let $t_1 = (x_1, y_1)$ be a translation which minimizes $f(t_1)$, the exact measure. Then $F[x_0, y_0]$ differs from $f(t_1)$ by at most 1, when the norm used, $\|\cdot\|$, is any L_p norm.*

For a proof of this claim see Appendix C. Note also that when $t = (x, y)$ is a translation with integer coordinates x and y, then $F[x, y] = f(t)$: the rasterized function is simply a sampling of the exact function.

Thus the minimum value of the rasterized approximation, $F[x, y]$, specifies the minimum Hausdorff distance under translation to an accuracy of one unit of quantization. However, it should be noted that the *translation* minimizing $F[x, y]$ is not necessarily close to the translation t which actually minimizes $f(t)$. To see an example of this, let k be any odd number, and let A be the set $\{(0,0),(k,0),(2k+1,0)\}$. Let B be the set $\{(0,0),(2k+1,0)\}$. Then the translation t which minimizes $f(t)$ (in the exact case) is $(k/2, 0)$, with $M_T(A, B) = k/2$. In the rasterized case, however, $M_T(A, B) = (k+1)/2$, with three translations which generate this minimum value: $((k+1)/2, 0)$, $((k-1)/2, 0)$ and $(-(k+1)/2, 0)$. Thus there there may be minimizing translations in the rasterized case that are arbitrarily far away from the minimizing translation in the exact case. This is not a problem, however, since the value of $H(A, B \oplus t)$ for each of these translations is the same (and the value at each translation must be within 1.0 of the exact minimizing value). In other words all three of these matches have the same cost in both the exact and rasterized cases, and this cost is nearly the exact minimum cost. In practice we generally enumerate all minimizing translations.

The function $f(t)$ and its rasterized approximation $F[x, y]$ specify the Hausdorff distance $H(A, B \oplus t)$ as a function of the translation t. The *directed* Hausdorff distance $h(B \oplus t, A)$ is also useful for comparing two bitmaps. In particular, in order to identify possible instances of a 'model' B in a cluttered 'image' A, it is often desirable to simply ensure that each portion of the model is near some portion of the image (but not necessarily vice versa). We denote by $F_B[x, y]$ the directed Hausdorff distance from B to A as a function of the translation (x, y) of B,

$$F_B[x, y] = \max_{b \in B} D'[b_x + x, b_y + y]. \tag{7}$$

This measures the degree to which $B[k, l]$ resembles $A[k, l]$, for each translation (x, y) of B. When each nonzero pixel of $B[k+x, l+y]$ is near some nonzero pixel of $A[k, l]$ then the distance will be small. If on the other hand, some nonzero pixel of $B[k+x, l+y]$ is far from all nonzero pixels of $A[k, l]$ then the distance will be large. The directed distance from A to B is analogously given by $F_A[x, y] = \max_{a \in A} D[a_x - x, a_y - y]$. Note that $F[x, y]$ is simply the pointwise maximum of these two directed distance functions.

4.1 Computing the Voronoi surface array $D[x, y]$

There are many methods of computing a distance transform (or rasterized approximation to a Voronoi surface). In this section we summarize some of the approaches that we have used for computing the distance transform $D[x, y]$ of a binary array $E[x, y]$ (where we denote the nonzero pixels of $E[x, y]$ by the point set E).

One method of computing $D[x, y]$ is to use a local distance transform algorithm such as that in [5], [10] or [15]. In practice we use a two-pass serial algorithm that approximates the distance transform using a local mask to propagate distance values through the array (such as that of [5]). Better distance values can be obtained using a method such as that of [15] which produces distance transform values that are exact for the L_1 and L_∞ norms, and are exact up to the machine precision for the L_2 norm. This algorithm first processes each row independently. For each row of $E[x, y]$, it calculates the distance to the nearest nonzero pixel in that row, i.e. for each (x, y) it finds Δx such that $E[x + \Delta x, y] \neq 0$ or $E[x - \Delta x, y] \neq 0$, and that Δx is the minimum non-negative value for which this is true. It then scans up and down each column independently, using the Δx values and a look-up table which depends on the norm being used, to determine $D[x, y]$.

Another method that we have used to compute distance transforms takes advantage of specialized graphics hardware for rendering and z-buffering. The form of $D[x, y]$ is, as noted in Section 2, an 'egg carton': the lower envelope of a collection of cone-shapes, one cone-shape for each point $e_j \in E$ (each nonzero pixel of $E[x, y]$), with its point at e_j. The exact form of the cone-shapes depends on the norm being used. For the L_1 norm the shapes are pyramids with sides of slope ± 1, oriented at 45° with respect to the coordinate axes. For the L_∞ norm they are again pyramids, but oriented parallel to the coordinate axes. For the L_2 norm they are cones of slope 1.

The computation of $D[x, y]$ is simply to take the pointwise minimum, or lower envelope, of the cone-shapes rendered as described above. Consider the operations performed by a graphics rendering engine set up to perform orthographic (rather than perspective) projection, and set up to view this collection of surfaces from 'below'. It can render the cones and perform visibility calculations quickly using a z-buffer. Suppose that location (x_0, y_0) in the z-buffer contains value d. Then the closest surface to the viewer which intersects the line $(x = x_0, y = y_0)$ is d away. This means that the lower envelope of the 'egg carton' is at height d at (x, y), and so $D[x, y] = d$. Thus we simply render each of the cones described above into a z-buffer doing orthographic projection (i.e., with a view from $z = -\infty$). The running time of this method is $O(p)$, where p is the number of points in the set E. This is because each source point results in the rendering of a single cone, and then the z-buffering operation is constant-time. With current graphics hardware tens of thousands of polygons per second can be rendered in a z-buffer, and thus it is possible to compute $D[x, y]$ in a fraction of second. For the L_1 and L_∞ norms, $D[x, y]$ is computed exactly; for the L_2 norm, there may be some error in the computed $D[x, y]$, which depends on the resolution of the z-buffer being used.

4.2 Computing the Hausdorff distance array $F[x, y]$

The Hausdorff distance as a function of translation, $F[x, y]$, defined in equation (6) can also be computed either using graphics hardware or standard array operations. For simplicity of discussion, we focus on the computation of the directed distance from the model to the image, $F_B[x, y]$ (the computation of $F_A[x, y]$ is analogous). Recall that $F[x, y]$ is just the maximum of these two directed distances.

Above we saw that $F_B[x, y]$ can be defined as the maximum of those values of $D'[x, y]$ (the distance transform of $A[k, l]$) that are selected by elements of B for each translation (x, y) of B. Alternately, this can be viewed as the maximization of $D'[x, y]$ shifted by each location where $B[k, l]$ takes on a nonzero value,

$$F_B[x, y] = \max_{b \in B} D'[b_x + x, b_y + y] = \max_{k,l; B[k,l]=1} D'[k + x, l + y]. \tag{8}$$

This maximization can be performed very rapidly with special-purpose graphics hardware for doing pan and z-buffer operations. We simply pan $D'[x, y]$ and accumulate a pointwise maximum (upper envelope) using a z-buffer. In practice, for most current graphics hardware this operation is not fast because it involves repeatedly loading the z-buffer with an array from memory.

A second way of computing $F_B[x, y]$, using standard array operations, arises from viewing the computation slightly differently. Note that (8) is simply equivalent to maximizing the product of $B[k,l]$ and $D'[x,y]$ at a given relative position,

$$F_B[x,y] = \max_{k,l} B[k,l]D'[k+x, l+y]. \tag{9}$$

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

In order to compute $F_B[x,y]$ using the method of equation (9), the array $B[k,l]$ is simply positioned centered at each pixel of the distance transform $D'[x,y]$. The value of $F_B[x,y]$ is then the maximum value obtained by multiplying each entry of $B[k,l]$ by the corresponding entry $D'[k+x,l+y]$. As $B[k,l]$ is just a binary array, this amounts to maximizing over those entries of $D'[k+x,l+y]$ that are selected by the nonzero pixels of $B[k,l]$. That is, we can view the nonzero model pixels as *probing* locations in the Voronoi surface of the image, and then $F_B[x,y]$ is the maximum of these probe values for each position (x,y) of the model $B[k,l]$.

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

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

The only differences are that the array $A[k,l]$ in the correlation is replaced by the distance array $D'[x,y]$ in equation (9) (the distance to the nearest pixel of $A[k,l]$), and the summation operations in the correlation are replaced by maximization operations. It should be noted that the directed Hausdorff distance is very insensitive to small errors in pixel locations, because the Voronoi surface, $D'[x,y]$, reports the distance to the *nearest* point of $A[k,l]$. Thus if pixels are slightly perturbed, the value of $A[k,l]$ only changes a small amount. In binary correlation, however, there is no such notion of spatial proximity. Either pixels are directly superimposed or not. Binary correlation is one of the most commonly used tools in image processing, and we will further examine the relation between our method and correlation in Section 6.

4.3 Matching portions of the model and image

We can use the partial distance $H_{LK}(A, B \oplus t)$, given in equation (5), to define a version of $F[x,y]$ which allows *portions* of a model and image to be compared. For a given translation $t = (x,y)$, and fractions f_1 and f_2 representing the fraction of the nonzero model and image pixels to be considered, respectively, let $K = \lfloor f_1 q \rfloor$ and $L = \lfloor f_2 p \rfloor$, and redefine

$$\begin{aligned} F_B[x,y] &= H_K(B \oplus t, A) = \underset{b \in B}{K^{\text{th}}} D'[b_x + x, b_y + y] \\ F_A[x,y] &= H_L(A, B \oplus t) = \underset{a \in A}{L^{\text{th}}} D[a_x - x, a_y - y] \\ F[x,y] &= H_{LK}(A, B \oplus t) = \max(F_A[x,y], F_B[x,y]). \end{aligned}$$

When $f_1 = f_2 = 1$, then $K = q$ and $L = p$, and this is the same as the old version of $F[x,y]$.

When we are considering the partial distance between an image and a model, the model is often considerably smaller than the image, reflecting the fact that in many tasks a given instance of the model in the image will occupy only a small portion of the image. In this case, the above definition of partial distance is not ideal for the directed distance from the image to the model, $H_L(A, B \oplus t)$, because we must define L, the number of image pixels that will be close to model pixels. This number, L, however will depend on how many objects are in the image.

A natural way to compute a partial distance from the image to the model is to consider only those image points that are near the current hypothesized position of the model, since those farther away are probably parts of other imaged objects. In practice, it is sufficient to consider only the image pixels which lie 'underneath' the current position of the model: if we are computing $F[x,y]$ and the model is m pixels by n pixels, then we compute a different version of $F_A[x,y]$ that considers just the points of $A[k,l]$ that are under the model at its given position, $B[k+x, l+y]$:

$$F_A[x,y] = \max_{\substack{(k,l) \\ x \leq k < m+x \\ y \leq l < n+y}} A[k,l] D[k-x, l-y]. \tag{10}$$

Note, that given this definition of comparing just a portion of the image to the model, it is possible to further compute the 'partial distance' of this portion of the image against the model. This can be done by combining this definition with the ranking-based partial distance. We can do this by adjusting the value of L depending on how many image pixels lie under the model at its given position (because we are computing a partial distance with just this portion of the image). In other words we let $L = \lfloor f_2 r \rfloor$, where r is the number of nonzero image pixels which are 'underneath' the translated model at the current translation. For this, the definition of $F_A[x,y]$ in equation (10) would be modified to use L^{th} instead of max.

5 Efficient Computation of $F[x, y]$

The naïve approach to computing $F[x, y]$, described in equation (6) of Section 4, can take a significant time to run, because it considers every possible translation of the model within the given ranges of x and y. We have developed some 'pruning' techniques that decrease this running time significantly. These techniques take advantage of the fact that, in typical applications, once $F[x, y]$ has been computed, it will generally be scanned to find all entries which are below some threshold, τ. We can use this to generate the (x, y) values where $F[x, y] \leq \tau$ directly, without generating all of $F[x, y]$. We present here some of the techniques which we have found to be useful.

The effects of these speedup techniques vary depending on the image and the model being used. They are more effective when the image is sparse, and when the model has a large number of points. In our work we have seen speedups of a factor of 1000 or more over the naïve approach. Some image/model pairs take only fractions of a second to compare (some illustrative timings will be presented with the examples below).

5.1 Ruling out circles

We wish to compute the Hausdorff distance as a function of translation, $F[x, y]$, given a binary 'image' $A[k, l]$ and a 'model' $B[k, l]$. Let the bounds on A be $0 \leq k < m_a$, $0 \leq l < n_a$, and the bounds on B be $0 \leq k < m_b$ and $0 \leq l < n_b$. While the array $F[x, y]$ is in principle of infinite extent, its minimum value must be attained when the translated model overlaps the image in at least one location, so we only consider the portion where $-m_b + 1 \leq x < m_a$ and $-n_b + 1 \leq y < n_a$.

One property of $F_B[x, y]$ is that its slope cannot exceed 1. That is, the function does not decrease more rapidly than linearly. Thus if $F_B[x_1, y_1] = v$ (where $v > \tau$) then $F_B[x, y]$ cannot be less than τ in a circle of radius $v - \tau$ about the point (x_1, y_1). (The actual shape of the "circle" depends on the norm used; it is a true circle for L_2). In other words, if the value of $F_B[x_1, y_1]$ is large at some location, then it cannot be small in a large area around that location. This fact can be used to rule out possible translations near (x_1, y_1). More formally,

Claim 4 *Let (x_1, y_1) and (x_2, y_2) be translations, with $-m_b < x_1, x_2 < m_a$ and $-n_b < y_1, y_2 < n_a$. Then $|F_B[x_1, y_1] - F_B[x_2, y_2]| \leq \|(x_1, y_1) - (x_2, y_2)\|$. This is true for all values of f_1 (the fraction of nonzero model pixels considered).*

For a proof of this claim, see Appendix D. We use this fact in the algorithm detailed below in order to speed up the computation.

This property does not necessarily hold for $F_A[x,y]$. If we are considering only the portion of the image under the model, as in Subsection 4.3, we might have a location where moving the model by one pixel 'shifts' some image points into or out of the window, which can change the value of $F_A[x,y]$ by a large amount. This also implies that the property does not necessarily hold for $F[x,y]$. In practice, however, this is not much of an issue because generally it is only for the image array that we wish to skip over parts of a large array (e.g., by ruling out circles). The model array is usually small enough that we do not need this technique.

5.2 Early scan termination

We may also obtain a speedup by not computing $F_B[x,y]$ completely if we can deduce partway through the computation that it will be greater than τ. Recall that F_B is computed by maximizing over all the locations of $D'[k+x,l+y]$ that are 'selected' by nonzero pixels of $B[k,l]$. That is, each nonzero pixel of $B[k,l]$ in effect *probes* a location in the Voronoi surface of $A[k,l]$, and we maximize over these probe values. Thus if a single probe value is over the threshold τ at translation (x,y), then we know that $F[x,y]$ must be over τ (it is the maximum over all the probe values). Thus we can stop computing F_B for this translation, because it is over threshold.

An analogous result holds for the partial distances. Let $K = \lfloor f_1 q \rfloor$. The value of $H_K(B \oplus (x,y), A)$ is the K-th ranked value of $D'[b_x+x, b_y+y]$, taken over all (b_x,b_y) where $B[b_x,b_y]=1$ (there are q such locations). We probe D' in q places, and maintain a count of the number of these values from D' which exceed τ. If this count exceeds $q-K$, we know that the K-th ranked value must be greater than τ, and so $F_B[x,y] > \tau$, so we need probe no more locations for the translation (x,y). In fact, we can determine the minimum possible value $F_B[x,y]$ could have at this location, by assuming that the unprobed values are all 0 and calculating the K-th ranked value of this set of values. We can use this to eliminate nearby values of (x,y) from consideration. This method works best for large values of f_1.

5.3 Skipping forward

A third technique relies on the order in which the space of possible translations is scanned. We must be scanning the distance transform array in some order; assume that the order is a row at a time, in the increasing x direction. In other words, for some y, we first consider $F_B[-m_b+1, y]$, then $F_B[-m_b+2, y]$, up to $F_B[m_a-1, y]$. In this case, it is possible to quickly rule out large sections of this row by using a variant of the distance transform.

Let $D'_{+x}[x, y]$ be the distance in the increasing x direction to the nearest location where $D'[x, y] \leq \tau$, and ∞ if there is no such location (in practice, $D'_{+x}[x, y]$ would be set to a large value if there is no such location; a value greater than the width of the array is sufficiently large). Formally,

$$D'_{+x}[x, y] = \min_{\substack{\Delta x \geq 0 \\ D'[x+\Delta x, y] \leq \tau}} \Delta x$$

Note that $D'_{+x}[x, y] \geq D'[x, y] - \tau$. We can use $D'_{+x}[x, y]$ to determine how far we would have to move in the increasing x direction to find a place where $F_B[x, y]$ might be no greater than τ. Let

$$G_B[x, y] = \underset{b \in B}{K^{\text{th}}} \, D'_{+x}[b_x + x, b_y + y]$$

If $G_B[x, y]$ is 0, then K of the values of D'_{+x} probed must have been 0, and so K of the values of D' which would be probed in the computation of $F_B[x, y]$ would be $\leq \tau$. Further, if $G_B[x, y] = \Delta x > 0$, then not only do we know that $F_B[x, y] > \tau$, but also $F_B[x+1, y], \ldots, F_B[x + \Delta x - 1, y] > \tau$. (The proof of this is similar to the proof of claim 4 and is omitted). We can therefore immediately increment x by Δx, and skip a section of this row. Note also that we do not need to compute $F_B[x, y]$ at all if we compute $G_B[x, y]$ and find that it is nonzero. Early scan termination can be applied to this computation.

This method has the advantage over ruling out circles that it does not require any auxiliary data structures to be maintained; once $G_B[x, y]$ has been computed, x can be immediately incremented. The ruling out circles method must keep track of what translations have been ruled out, and updating this map can be time-consuming.

5.4 Interactions between speedup methods

These techniques may be used in combination with each other. However, using one technique may affect the efficiency of others. Interactions to be noted are:

- Using early scan termination greatly degrades the effect of both ruling out circles and skipping forward. Early scan termination will generally give a value for $F_B[x, y]$ which is only a small amount greater than τ, and so very few locations will be ruled out; continuing the scan could increase the value computed, thereby saving work later.

A possible solution for this is to terminate the scan when $F_B[x, y]$ has been shown to be greater than $\tau + R$, where R is some value, so that on a terminated scan, a circle of radius at least R could be ruled out; similarly, terminate the scan when $G_B[x, y]$ has been shown to be at least R. The value of R is arbitrary, and can be adjusted for best performance.

- The order in which translations are considered can be arbitrary if skipping forward is not used. The optimal order may well not consider adjacent translations successively, as this will tend to consider translations which are on the edges of ruled out circles, and so much of the neighborhood has already been ruled out. Considering a translation in a "clear" area would provide more opportunity for ruling other translations out.

5.5 An Efficient Algorithm

These observations give us an algorithm which will produce a list of values where $F[x, y] \le \tau$ quite efficiently.

Algorithm 1 *Given two input binary image arrays, $A[k, l]$ and $B[k, l]$, two fractions f_1, f_2, $0 \le f_1, f_2 \le 1$, and a threshold $\tau \ge 0$, generate a list T of (translation, value) pairs $((x, y), v)$ such that $v = F[x, y]$ and $v \le \tau$. Use the given fractions of $B[k, l]$ and $A[k, l]$ for each translation (x, y) of $B[k, l]$. Consider only a fraction of that part of $A[k, l]$ which is covered by the translated $B[k, l]$.*

1. Let the bounds of $A[k, l]$ be $0 \le k < m_a$ and $0 \le l < n_a$ and the bounds of $B[k, l]$ be $0 \le k < m_b$ and $0 \le l < n_b$.

2. Compute the array $D[x, y]$ that specifies the distance to the closest nonzero pixel of $B[k, l]$, making $D[x, y]$ the same size as $B[k, l]$.

3. Compute the array $D'[x, y]$ that specifies the distance to the closest nonzero pixel of $A[k, l]$, making $D'[x, y]$ with $-m_b < x < m_a + m_b - 1$ and $-n_b < y < n_a + n_b - 1$ (see Subsection 4.3).

4. Compute the array $D'_{+x}[x, y]$ that specifies the distance to the closest pixel (in the increasing x direction) of $D'[x, y]$ which is less than or equal to τ. Make $D'_{+x}[x, y]$ the same size as $D'[x, y]$.

5. Let the number of model pixels considered be $K = \lfloor f_1 q \rfloor$.

6. Create an array, $M[x, y]$, the same size as $D'[x, y]$. This will contain the minimum possible value that $F_B[x, y]$ could have, given the information we have accumulated. Initialize $M[x, y]$ to zero.

7. Create two lists, T and T'. Initialize both to empty.

8. For each translation, let the number of image pixels considered be $L = \lfloor f_2 r \rfloor$, where r is the number of nonzero pixels in $A[k, l]$ that are covered by the current position of $B[k, l]$.

9. For each translation (x, y) of $B[k, l]$ (scanned in reading order: top to bottom, left to right),

   (a) If $M[x, y] > \tau$, then we need not consider this translation at all, and should proceed to the next one. Otherwise,

   (b) Set o to zero.

   (c) For each $b \in B$, consider $D'_{+x}[b_x + x, b_y + y]$. If it is greater than R, increment o. Also consider $D'[b_x + x, b_y + y]$. If, during this process, o exceeds $q - K$, then

      i. Take the smallest of the D'_{+x} values which we have seen that exceeded 0. Call this Δx.

      ii. Take the smallest of the D' values which we have seen that exceeded τ. Call this v'.

      iii. For each value (x', y'):
      Set $M[x', y'] = \max(M[x', y'], v' - \|(x, y) - (x', y')\|)$. This only needs to be done for the $M[x', y']$ within a radius of $v' - \tau$ of (x, y), and need not be done at all for any (x', y') which has previously been considered in step 9.

      iv. Skip to the next translation, $(x + \Delta x, y)$ (if $x + \Delta x \geq m_a$, go to the start of the next row).

   (d) If o never exceeds $q - K$, then let v' be the K-th ranked value of the q values from D' generated in step 9c. This will be $\leq \tau$, since no more than $q - K$ of these values can be greater than τ. Add $((x, y), v')$ to the list T'.

10. The list T' now contains all the (translation, value) pairs $((x, y), v')$ such that $v' = F_B[x, y] \leq \tau$. For each $((x, y), v')$ on the list T'
    (a) Consider the values of $A[a_x, a_y]D[a_x - x, a_y - y]$, for all points $a \in A$ such that $x \leq a_x < x + m_b$ and $y \leq a_y < y + n_b$; compute the L-th ranked value. (Recall that $L = \lfloor f_2 r \rfloor$ where r is the number of points of A that lie under $B[k, l]$ for the current translation (x, y), as described in Subsection 4.3.) Call this value v. We know that $F[x, y] = \max(v', v)$.
    (b) If v is less than or equal to τ, add $((x, y), \max(v', v))$ to the list T.

This algorithm can also be used to produce a list of translations where the directed Hausdorff 'distance' from the model to the image is less than τ, by halting after step 9 and using the list T'.

6 Examples

We now consider some examples in order to illustrate the performance of the Hausdorff distance methods developed above, using some image data from a camera in our laboratory. The first test image is shown in Figure 3. This binary image is 360×240 and was produced by applying an edge operator (similar to [6]) to a grey-level camera image. The computation of the Hausdorff distance under translation was done using an implementation written in C of the algorithm described above.

The model to be compared with the first test image is shown in Figure 4. The outline around the figure delineates the boundary of the bitmap representing the model. The model is 115×199 pixels. Comparing this model against this image with $\tau = 2$ pixels, $f_1 = 0.8$ and $f_2 = 0.5$ takes approximately twenty seconds on a Sun-4 (SPARCstation 2). This produced two matches, at $(87, 35)$ and $(87, 36)$. Figure 5 shows the match at $(87, 36)$ overlaid on the image. As a comparison, the naïve approach from Subsection 4.2 takes approximately 5000 seconds to perform this comparison process.

We also ran the algorithm on the image and model shown in Figures 6 and 7, using $\tau = 1.42$, $f_1 = 0.66$ and $f_2 = 0.35$. The image is 256×233 and the model is 60×50. Four matches were found, at $(99, 128)$, $(100, 128)$, $(99, 129)$ and $(100, 129)$. Figure 8 shows the match at $(99, 129)$ overlaid on the image. The computation took approximately 5 seconds.

Our third test case consists of the image and model shown in Figures 9 and 10, using $\tau = 2.83$, $f_1 = 1$ and $f_2 = 0.5$. The image is 360×240 and the model is 38×60. The

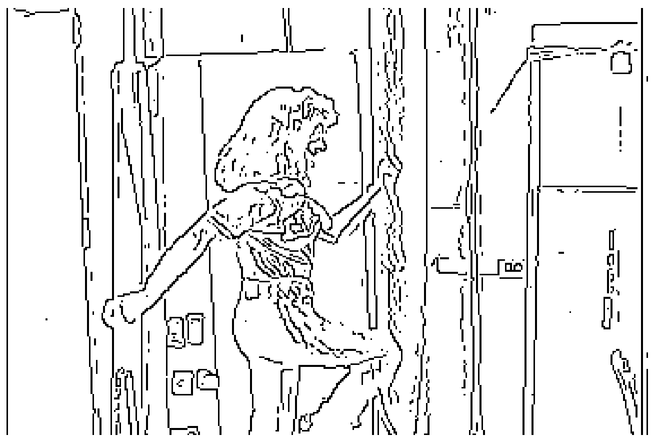

Figure 3: The first test image.

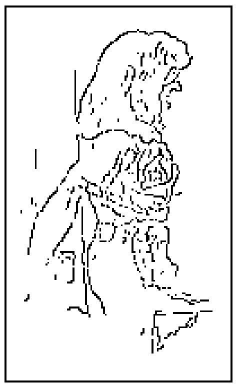

Figure 4: The first object model.

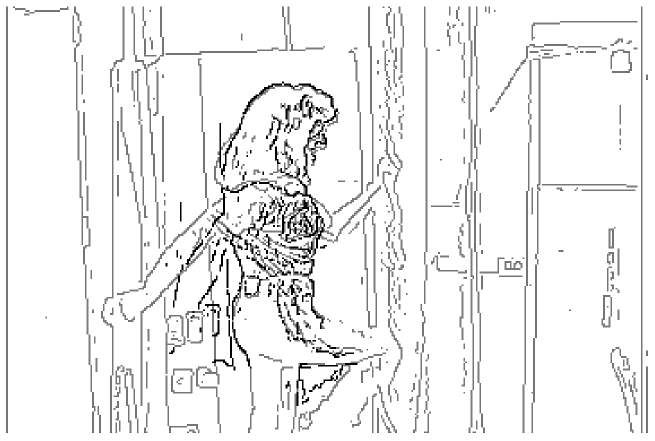

Figure 5: The first test image overlaid with the best match.

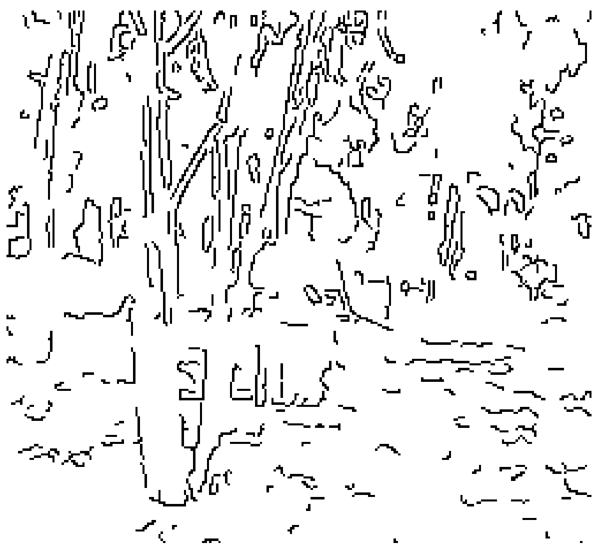

Figure 6: The second test image.

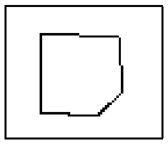

Figure 7: The second object model.

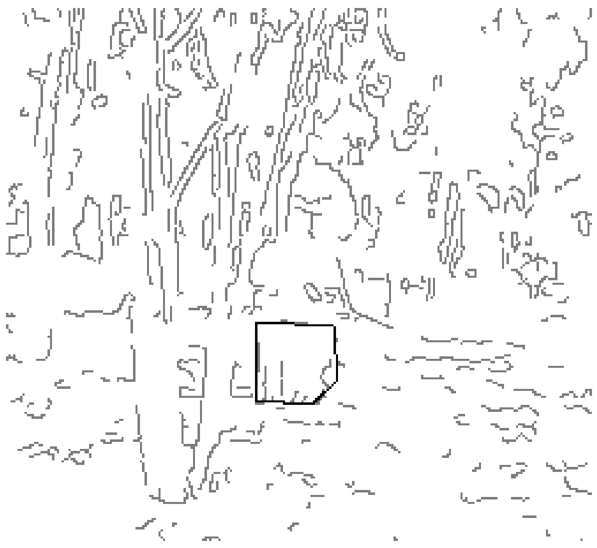

Figure 8: The second test image overlaid with the best match.

model was digitized from a different can, held at approximately the same orientation and same distance from the camera. Four matches were found, at $(199, 95)$, $(199, 98)$, $(200, 98)$ and $(199, 99)$. Figure 11 shows the match at $(199, 95)$ overlaid on the image. The computation took approximately 1.4 seconds.

If several models are to be compared against the same image, then the distance transform of the image need only be computed once. Our implementation takes about one second to compute the distance transform of a 256×256 image on a Sun-4 (SPARC-station 2). Once this has been computed, the comparisons can take as little as one half second per model (256×256 images, 32×32 model). The time taken depends on the τ, f_1 and f_2 values used; larger τ and smaller f_1 values increase the time taken. A more cluttered image will also increase the time.

In order to compare the directed Hausdorff distance with correlation, we computed the correlation of the stump model in Figure 7 with the second test image. We defined a 'match' to be a translation where the correlation function was at a local peak. For this image, the correlation performed poorly. There were eight incorrect matches which had a higher correlation value than the correct match, and the correct match had a peak value of only 77% of the largest peak. None of the incorrect matches was close (spatially) to the correct match.

Hence we see support from these examples for our theoretical claim that the partial Hausdorff 'distance' works well on images where the locations of image pixels have been perturbed. Moreover, these same images cannot be handled well by correlation.

7 The Hausdorff Distance Under Rigid Motion

The methods we have described for computing the Hausdorff distance under translation can be naturally extended to computing the Hausdorff distance under rigid motion (translation and rotation). In this case, we require that the norm used be the Euclidean norm (L_2). As before, we fix the set A and allow the set B to move (in this case rotate and translate). The minimum value of the Hausdorff distance under rigid (Euclidean) motion, $M_E(A, B)$, then gives the best transformation of B with respect to A,

$$M_E(A, B) = \min_{t,\theta} H(A, (R_\theta B) \oplus t) \tag{11}$$

where H is the Hausdorff distance as defined in equation (1), $(R_\theta B) \oplus t = \{R_\theta b + t \mid b \in B\}$, and R_θ is the standard rotation matrix. This distance is small exactly when there

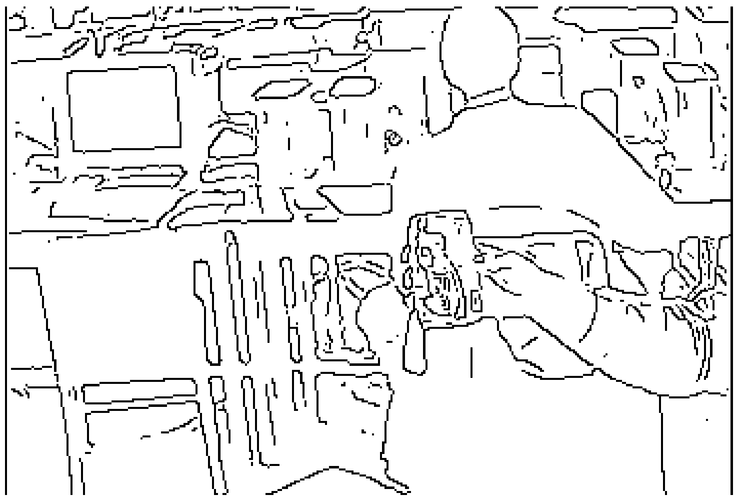

Figure 9: The third test image.

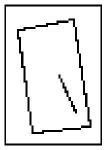

Figure 10: The third object model.

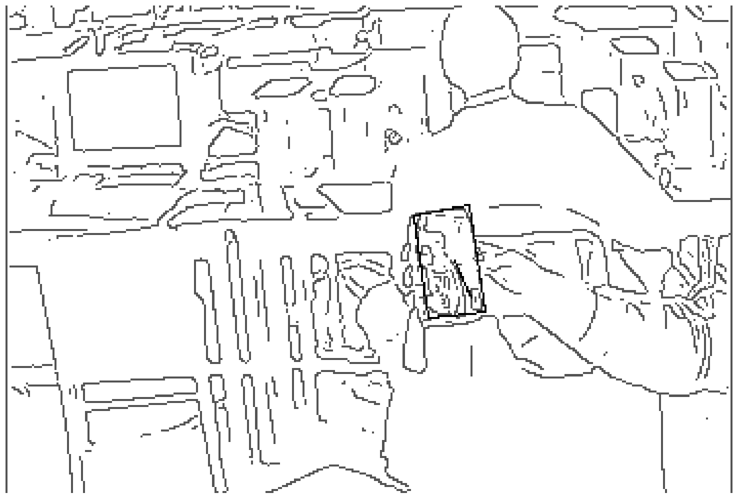

Figure 11: The third test image overlaid with the best match.

is a Euclidean transformation that brings every point of B near some point of A and vice versa.

We can compute a rasterized approximation to the minimum Hausdorff distance under rigid motion. As above, the basic idea is to compute the Hausdorff distance for all transformations of the model at the appropriate level of rasterization, and then find the minima for which the distance is below some threshold, τ. For translations, the appropriate level of rasterization is again single pixels. For rotation, we want to ensure that each consecutive rotation moves each point in the model by at most one pixel. Each point b_j in the set B is being rotated around the center of rotation, c_r, on a circle of radius $r_j = \|b_j - c_r\|_2$. This means that the rasterization interval in θ, $\Delta\theta$, should be $\arctan(1/r_k)$ where r_k is the radius of rotation of the point in B which is furthest from the center of rotation.

Our current implementation is restricted to computing only the directed Hausdorff distance from the model, B, to the image, A. This is analogous to $F_B[x, y]$ as defined in equation (7). Furthermore, we consider only complete shapes (no partial distances). Recall that the Hausdorff distance computation can be thought of as probing locations in the Voronoi surface of the image corresponding to the transformed model points. The probe values for each transformation are then maximized in order to compute the distance for that transformation. For rigid motion, we build a structure which gives for each model point a list of the locations that the point moves through as it rotates, one location for each rasterized θ. These locations are relative to the center of rotation and, for each model point, this list describes a circle about the center of rotation.

Consider the problem of determining the minimum (directed) Hausdorff distance under rotation for a fixed translation t, $\min_\theta h((R_\theta B) \oplus t, A)$. We first initialize to zero an array, Q, containing an element for each θ value. During the computation, this array will contain the minimum possible value of the Hausdorff distance for each rotation, on the basis of the points that have been probed thus far. For each point we probe the distance transform of the image at each relative rotated location, and maximize these probe values with the corresponding values in the array Q. Once all points in B have been considered, the array Q gives the directed Hausdorff distance for each discrete rotation at the current translation:

$$Q[i] = h((R_{i\Delta\theta} B) \oplus t, A)$$

We perform this computation for each rasterized translation t. This algorithm is analogous to the naïve algorithm for the translation-only case.

As in the translation-only case, many possible translations and rotations of B may be ruled out without explicitly considering them. We have begun to investigate speedup techniques for Euclidean motion and now describe the methods that have proven successful.

First, we choose as the center of rotation that point of B which is closest to the centroid of the model. This both reduces the number of rasterized θ values which we need to consider and also explicitly makes the center of rotation be a point in B. This is useful since the center of rotation does not move as θ changes. Thus, if at a given translation we probe the center point first and find that the distance transform at that point is greater than τ, then we know that there are *no* good rotations of the model at this translation.

Points of B that are close to the center of rotation will not pass through many distinct grid points as they rotate about it. In building the model rotation structure, we can therefore store only the distinct grid points through which each point of B rotates. In addition, we also store the range of rasterized θ values that each of these distinct points corresponds to. This compression of the rotation structure eliminates a large number of extraneous probes. However, a single probe may now be used to update several consecutive entries in Q.

Techniques analogous to ruling out circles and early scan termination can also be used in θ-space to prune the search for good transformations of the model.

If we find that for some rotation θ and point $b \in B$ the distance from $R_\theta b + t$ to the nearest point in A is v and $v > \tau$, then all rotations which bring b into the circle of radius $v - \tau$ centered at $R_\theta b + t$ may be eliminated from consideration. These are the rotations in the range $\theta \pm \cos^{-1}(1 - (v - \tau)^2/(2r^2))$, where r is the radius of rotation of point b. If $v - \tau > 2r$ then all values of θ may be ruled out. Note that a conservative approximation to this range is $\theta \pm (v - \tau)/r$, which may be used if $\cos^{-1}$ is too expensive to compute. Considering the points in order of increasing radius of rotation makes it likely that large θ ranges will be eliminated early on, as a single large probe value for one of the "inner" points will rule out more θ values than it would for one of the "outer" points.

By maintaining a list of ruled-out regions of θ-space, the transformations resulting in a Hausdorff distance greater than τ can be quickly discarded. As noted in subsection 5.4, combining early scan termination with ruling out circles for the translation-only case may result in only small areas being ruled out. This same problem holds in θ-space, and again, a possible solution is to delay early termination until we can guarantee that the terminated scan eliminates at least some minimum radius circle.

It is also possible to use pruning techniques in both rotation and translation simultaneously. One possible technique is to use the Q array generated for one translation to eliminate some θ values for an adjacent translation. Since the slope of the distance transform cannot exceed 1, in moving to an adjacent translation each probe value could at worst decrease by 1. Thus, any angle for which the lower bound on the Hausdorff distance is greater than $\tau + 1$ at the current translation can also be ruled out at all adjacent translations. In practice, however, we found that the extra overhead involved actually slowed down the computation.

Finally, to illustrate the performance of our current implementation, we use an image of some children's blocks on a table. This example is taken from a demonstration in which a robot arm locates blocks on the table using the Hausdorff distance under rigid motion and then uses the blocks to build an object the user has specified. Figure 12 shows the edge detector output for the 360×240 grey-level camera image of the blocks. The block model is shown in Figure 13 and is simply a square 31 pixels on each side. Matching this model against the image using $\tau = 3$ pixels, 60 local minima are found below threshold, corresponding to four rotations for each of the 15 blocks which are completely in the field of view. The matches are shown overlaid on the original image in Figure 14. Note that because we are only computing the directed distance from the model to the image, even the blocks that contain letters imprinted in the upward facing side are recognized. At each translation and rotation of the model, we merely require that there is an image point within three pixels of each point in the transformed model. This is exactly what is needed for this application, though large black areas in the image would present problems because spurious matches would be found. This matching takes approximately 216 seconds on a SPARCstation 2. Taking advantage of the symmetry of this particular model would allow a four-fold increase in speed for this application.

With more time spent optimizing our pruning techniques, the running time should be greatly improved, especially considering the improvement achieved over the naïve algorithm for the translation-only case.

8 Summary

The Hausdorff distance under translation measures the extent to which each point of a translated 'model' set lies near some point of an 'image' set and vice versa. Thus this

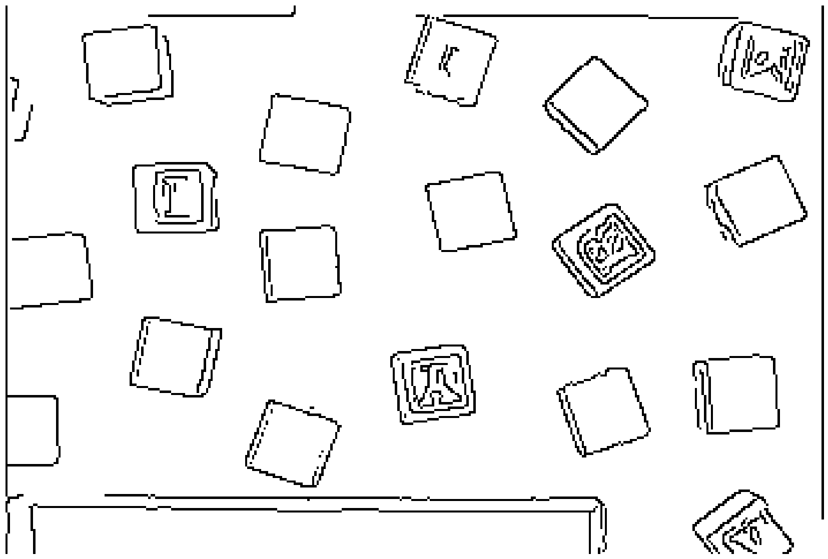
Figure 12: A test image showing some blocks.

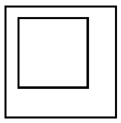
Figure 13: The block model.

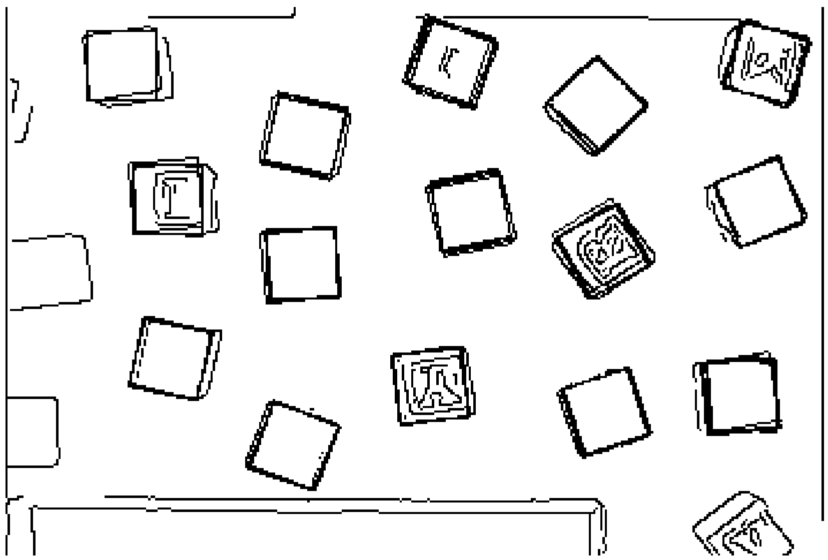
Figure 14: The test image blocks overlaid with each matched block.

distance reflects the degree of resemblance between two objects (under translation). We have discussed how to compute the Hausdorff distance under translation efficiently for binary image data. The method compares a 32×32 model bitmap with a 256×256 image bitmap in a fraction of a second on a SPARCstation 2.

The computation of the directed Hausdorff distance under translation is in many ways similar to binary correlation. The method is more tolerant of perturbations in the locations of points than correlation because it measures proximity rather than exact superposition. This is supported by empirical evidence as well as by the theoretical formulation of the problem. The *partial* Hausdorff 'distance' between a model and an image has been illustrated to work well on examples where correlation fails. We have also extended our algorithms to work with rigid motion. It is an open problem to develop efficient methods, both theoretically and in practice, for computing the Hausdorff distance under other transformation groups.

A Proof of Claim 1

Proof. Let A' be the points in A for which there is some point in B within d:

$$A' = \{a \in A \,|\, \exists b \in B \text{ such that } \|a - b\| \leq d\}.$$

Similarly, let B' be the points in B for which there is some point in A within d. We must have $|A'| \geq L$ and $|B'| \geq K$, since $h_L(A, B) = L^{\text{th}}{}_{a \in A} \min_{b \in B} \|a - b\| \leq d$ implies that there are at least L points in A which are closer than d to some point in B (and similarly there are K points in B which are closer than d to some point in A). Also, since $\| \cdot \|$ is symmetrical (i.e. $\|a - b\| = \|b - a\|$), all the 'neighbors' of any point in A' must be in B' and vice versa.

The problem now reduces to finding $A_L \subseteq A'$ and $B_K \subseteq B'$ having $\min(K, L)$ points each such that $H(A_L, B_K) \leq d$. We will show that this is possible by building A_L and B_K one element at a time, while maintaining the invariant that $H(A_L, B_K) \leq d$ (i.e. that for each element of A_L there is some element of B_K within d and vice versa).

Base case: Pick any point a from A'. Put it into A_L. Find any point b in B' such that $\|a - b\| \leq d$. There must be at least one. Put b into B_K. Then $H(A_L, B_K) = \|a - b\| \leq d$.

Induction step: Suppose that A_L and B_K each have $n < \min(K, L)$ elements and that $H(A_L, B_K) \leq d$. There are now two cases:

- Suppose that there exist $a \in A' - A_L$ and $b \in B' - B_K$ such that $\|a - b\| \leq d$. Then if we add a to A_L and b to B_K we will have increased the size of each set to $n + 1$ while maintaining the invariant.
- Suppose that no such a and b exist. Then pick any point $a \in A' - A_L$ and consider its 'neighbors': points in $B' \oplus t$ within d. It must have at least one since it is a member of A'. All the neighbors must be in B_K already, so it has at least one neighbor in B_K. Similarly, every point $b \in B' - B_K$ has at least one neighbor in A_L. Picking any point in $A' - A_L$ and any point in $B' - B_K$ and adding them to A_L and B_K respectively increases the size of each set to $n + 1$ and maintains the invariant, since every point in the new A_L has a neighbor in the new B_K and vice versa.

Since there are at least L elements in A' and K elements in B', we will not run out of elements in $A' - A_L$ and $B' - B_K$ before achieving $\min(K, L)$ elements in A_L and B_K.

This process builds A_L and B_K by ensuring that whenever a pair of points are added, they will each have neighbors in the augmented sets, which maintains the desired invariant. ∎

B Proof of Claim 2

Proof. We will be using A', B', A_L and B_K from the proof of Claim 1 to construct A'_L and B'_K. Suppose (without loss of generality) that $K \geq L$. Now, pick any $K - L$ points from $B' - B_K$ (there must be at least this many points since $|B_K| = \min(K, L) = L$ and $|B'| \geq K$). Let B'_K be the union of B_K and these points. For each of the new points, pick one of its neighbors from A'. Let A'_L be the union of A_L and these neighboring points. $|A'_L|$ will be at most K and at least L, since $|A_L| = L$, and all these neighbors might have been members of A_L. Thus, $L \leq |A'_L| \leq K = \max(K, L)$. Since every point in B'_K has a neighbor (within d) in A'_L and vice versa, we know that $H(A'_L, B'_K) \leq d$. However, we must have equality since if $H(A'_L, B'_K)$ were strictly less than d, then $H_{LK}(A, B)$ would also be strictly less than d, since A'_L and B'_K would then be minimizing subsets of sizes at least L and K respectively. ∎

C Proof of Claim 3

Proof. We can see from the definitions of D and D' that if x and y are integers, and $t = (x, y)$, then $D[x, y] = d(t)$ and $D'[x, y] = d'(t)$. Thus, $F[x, y] = f(t)$, and so the minimum value of $F[x, y]$ is no smaller than the minimum value of $f(t)$: $F[x_0, y_0] \geq f(t_1)$.

Since $\| \cdot \|$ is a norm, it satisfies the triangle inequality. Let a be any point. $d(a)$ is the distance from a to the nearest point of B, and so there is a point b in B such that $\|b - a\| = d(t)$. Let a' be any other point. Then

$$d(a') \leq \|b - a'\| \leq \|b - a\| + \|a - a'\| = d(a) + \|a - a'\|.$$

Similarly, $d'(b') \leq d'(b) + \|b - b'\|$ for any two points b and b'.

If t and t' are any two translations, then $d(a - t') \leq d(a - t) + \|t - t'\|$ and $d'(b + t') \leq d(b + t) + \|t - t'\|$. Then

$$f(t') = \max \left(\max_{a \in A} d(a - t'), \max_{b \in B} d'(b + t') \right)$$

$$\begin{aligned} &\leq \max\left(\max_{a\in A} d(a-t) + \|t-t'\|, \max_{b\in B} d'(b+t') + \|t-t'\|\right) \\ &= f(t) + \|t-t'\| \end{aligned}$$

Let $t'_1 = (x'_1, y'_1)$ be the grid point closest to t_1: x'_1 and y'_1 are the integers which minimize $\|t'_1 - t_1\|$. Since $\|\cdot\|$ is an L_p norm,

$$\|t'_1 - t_1\| \leq \|t'_1 - t_1\|_1 = |x_1 - x'_1| + |y_1 - y'_1| \leq 1.$$

$F[x_0, y_0] \leq F[x'_1, y'_1]$ since (x_0, y_0) minimizes $F[x,y]$. Thus,

$$f(t_1) \leq f(t_0) = F[x_0, y_0] \leq F[x'_1, y'_1] = f(t'_1) \leq f(t_1) + \|t_1 - t'_1\| \leq f(t_1) + 1.$$

∎

D Proof of Claim 4

Proof. From the proof of claim 3, we know that

$$D'[x_2, y_2] \leq D'[x_1, y_1] + \|(x_1, y_1) - (x_2, y_2)\|.$$

Let $v_1 = F_B[x_1, y_1]$ and $v_2 = F_B[x_2, y_2]$. Suppose that $v_1 \leq v_2$. We know that $K = \lfloor f_1 q \rfloor$ of the nonzero model pixels are within v_1 of some nonzero image pixel: there exist $(bx_1, bx_1), \ldots, (bx_K, bx_K)$ such that $D'[b_x + x_1, b_x + y_1] \leq v_1$, for $1 \leq j \leq K$.

Now consider the computation of v_2. We will be 'probing' the values of $D'[b_x + x_2, b_x + y_2]$ for $1 \leq j \leq K$ (among others). But since

$$\begin{aligned} D'[b_x + x_2, b_x + y_2] &\leq D'[b_x + x_1, b_x + y_1] + \|(x_1, y_1) - (x_2, y_2)\| \\ &\leq v_1 + \|(x_1, y_1) - (x_2, y_2)\|, \end{aligned}$$

there are at least K nonzero model pixels which are at most $v_1 + \|(x_1, y_1) - (x_2, y_2)\|$ away from some nonzero image pixel when the model is translated by (x_2, y_2). Since the computation of v_2 computes the K-th ranked value of these distances, we will have $v_2 \leq v_1 + \|(x_1, y_1) - (x_2, y_2)\|$. ∎

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[2] H. Alt, B. Behrends, and J. Blomer. Measuring the resemblance of polygonal shapes. In *Proc. Seventh ACM Symposium on Computational Geometry*, 1991.

[3] E. 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 Analysis and Machine Intelligence*, 13(3):209–216, 1991.

[4] P.J. Besl and R.C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–154, 1985.

[5] G. Borgefors. Distance transforms in digital images. *IEEE Trans. Pat. Anal. and Mach. Intel.*, 34:344–371, 1986.

[6] J.F. Canny. A computational approach to edge detection. *IEEE Trans. Pat. Anal. and Mach. Intel.*, 8(6):34–43, 1986.

[7] L.P. Chew and K. Kedem. ****. ******, ****.

[8] R.T. Chin and C.R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, 1986.

[9] A. Csaszar. *General Topology.* Adam Hilger Ltd., Bristol, 1978.

[10] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[11] W.E.L. Grimson with T. Lozano-Pérez and D.P. Huttenlocher. *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, Cambridge, 1990.

[12] D.P. Huttenlocher and K. Kedem. Efficiently computing the Hausdorff distance for point sets under translation. In *Proceedings of Sixth ACM Symposium on Computational Geometry*, pages 340–349, 1990.

[13] D.P Huttenlocher, K. Kedem, and J.M. Kleinberg. On dynamic Voronoi diagrams and the minimum hausdorff distance for point sets under Euclidean motion in the plane. In *Proceedings of the Eighth ACM Symposium on Computational Geometry*, 1992. To appear.

[14] D.P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. In *Proceedings of Seventh ACM Symposium on Computational Geometry*, pages 194–293, 1991.

[15] D.W. Paglieroni. Distance transforms: Properties and machine vision applications. *Computer Vision, Graphics and Image Proc.: Graphical Models and Image Processing*, 54(1):56–74, 1992.

[16] F.P. Preparata and M.I Shamos. *Computational Geometry.* Springer-Verlag, New York, 1985.

[17] A. Rosenfeld and A. Kak. *Digital Picture Processing, 2nd ed.* Academic Press, New York, 1982.