A Multi-Resolution Technique for Comparing Images Using the Hausdorff Distance*

Daniel P. Huttenlocher
William J. Rucklidge

TR 92-1321
December 1992

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*This work was supported in part by National Science Foundation PYI grant IRI-9057928 and matching funds from General Electric, Kodak and Xerox, and by Air Force contract AFOSR-91-0328.

A Multi-Resolution Technique for Comparing Images Using the Hausdorff Distance *

Daniel P. Huttenlocher
William J. Rucklidge

Department of Computer Science
Cornell University
Ithaca, NY 14853
dph@cs.cornell.edu
wjr@cs.cornell.edu

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. In this paper we describe an efficient method of computing this distance, based on a multi-resolution tessellation of the space of possible transformations of the model set. We focus on the case in which the model is allowed to translate and scale with respect to the image. This four-dimensional transformation space (two translation and two scale dimensions) is searched rapidly, while guaranteeing that no match will be missed. We present some examples of identifying an object in a cluttered scene, including cases where the object is partially hidden from view.

*This work was supported in part by National Science Foundation PYI grant IRI-9057928 and matching funds from General Electric, Kodak and Xerox, and in part by Air Force contract AFOSR-91-0328.

1 Introduction

We present a method for efficiently computing the Hausdorff distance between all transformations of a model and an image (for some specified group of transformations). The Hausdorff distance is a max-min distance investigated in [8, 9] (and described below) for comparing point sets. Our method improves considerably on the techniques presented in those papers, and this improvement enables us to consider higher-dimensional transformation spaces. The primary focus in this paper is on transformations where a two-dimensional model is allowed to translate and scale (separately in x and y). This four-dimensional space of transformations can be useful in tasks where a two-dimensional image is taken of an object that is translating in the three-dimensional world. It is also useful in document recognition contexts, such as the interpretation of engineering drawings, where tokens are at an unknown location and scale, but at certain canonical orientations. We present examples for both of these applications.

The main advance of the work reported here is the development of a multi-resolution method for searching the space of possible transformations of a model with respect to an image. This allows large portions of the space to be ruled out relatively cheaply. The multi-resolution method is augmented with a random sampling strategy, which provides a heuristic estimate of whether or not a given region of the transformation space contains a "match" of the model to the image. By randomly selecting a relatively small number of computations to perform, the search is able to quickly determine whether or not a given region of the transformation space should be ruled out. This random sampling method will not miss reporting any matches (i.e., it is guaranteed not to omit a region where there is a match). The random sampling method may report matches where there are none, but it will seldom do so, and these false matches are ruled out in a final processing stage.

We thus obtain an efficient method for comparing sets of points, which applies to object recognition problems where a set of model features are to be compared with a set of image features. We illustrate the method with examples from the above-mentioned domains: intensity edges extracted from indoor scenes, and binary scans of engineering drawings (see Figures 1 and 2). These examples illustrate two advantages of the method. First it is relatively insensitive to small perturbations of the image. Second it naturally allows for *portions* of a model to be compared with portions of an image.

1.1 The Hausdorff Distance

Given two finite point sets $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$, 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. Throughout this paper we will use the L_2 or Euclidean norm (although the same results hold for other L_p norms).

The function $h(A, B)$ is called the *directed* Hausdorff distance from A to B. In effect, $h(A, B)$ ranks each point of A based on its distance to the nearest point of B, and then the largest ranked such point (the most mismatched point of A) specifies the value of the distance. 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). Note that in general $h(A, B)$ and $h(B, A)$ can attain very different values (the directed distances are not symmetric).

The computation of $H(A, B)$ does not involve determining an explicit pairing (or correspondence) of points of A with points of B (for example many points of A may be close to the *same* point of B). This contrasts with most model-based recognition methods (cf. [2, 4, 6]) which determine a correspondence between points of the model and the image.

While the Hausdorff distance does not allow for comparing portions of two sets A and B, because it measures the most mismatched points, there is a natural extension that measures the distance between some subset of A and some subset of B. We discuss this in Section 3.

The Hausdorff distance measures the mismatch between two sets that are at *fixed* positions with respect to one another, whereas we are interested in comparing models to images, where the models can be transformed by the action of some transformation group. Let A denote a set of image points (e.g., binary features of some kind) and B denote a set of model points. We will focus on the case where the model transformations are the group of translations and independent x and y scales; a transformation is thus a quadruple $t = (t_x, t_y, s_x, s_y)$, which maps a point $w = (w_x, w_y)$ into the point $t(w) = (s_x w_x + t_x, s_y w_y + t_y)$. We are then interested in the Hausdorff distance as a function of the transformation of the set B, or

$$f(t) = H(A, t(B)) \tag{3}$$

where H is the Hausdorff distance as defined in equation (1).

In addition we will use $f_B(t)$ to denote $h(t(B), A)$, the directed distance from the model to the image, which we call the *forward distance*, and $f_A(t)$ to denote $h(A, t(b))$ which we call the *reverse distance*. The natural asymmetry of the Hausdorff distance (i.e., the fact that the forward and reverse distances are generally not the same) can be thought of as corresponding to a hypothesize and test method. In other words, those values of t where the forward distance $f_B(t)$ is small are possible transformations of the model (hypotheses). For such values of t we test the hypothesis by determining whether the reverse distance $f_A(t)$ is also small. (Note that $f(t) = \max(f_B(t), f_A(t))$, and thus $f(t)$ is only small for values of t that "pass" both the hypothesis and the test.)

We now turn to the problem of efficiently computing $H(A, B)$ and $f(t)$. The organization of the remainder of the paper is as follows. We first discuss how to compute $H(A, B)$ for sets of points in the plane. The basic idea is to define functions measuring the distance from each point of A to the closest point of B (and vice versa), as a function of the transformation 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). In Section 4, we present a search strategy which operates at multiple resolutions, to efficiently find the values of t such that $f(t) = H(A, t(B))$ is small. Then in Sections 5 and 6 we discuss an implementation of the distance computation for raster data, where the sets A and B are represented in terms of binary arrays. In Section 7 we show how to improve the basic implementation, by ruling out many possible transformations of B without explicitly considering them. In Section 8 we then present some examples.

2 Computing $H(A, B)$ and $f(t)$

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

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

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 [7]. 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, as described in [11]). The graph of $d'(x)$ has a similar shape, with a "cone-shape" rising up from each point of A. For sets of grid points, the Voronoi surface, $d(x)$, of a set B has also been referred to as a *distance transform* (e.g., [5]), because it gives the distance from any point x on the grid to the nearest point in a set of source points, B.

We now turn to calculating the Hausdorff distance as a function of translation and scale. For a transformation $t = (t_x, t_y, s_x, s_y)$, from (2) we know that the forward distance is

$$\begin{aligned} f_B(t) = h(t(B), A) &= \max_{b \in B} \min_{a \in A} \|a - t(b))\| \\ &= \max_{b \in B} \min_{a \in A} \|a - (s_x b_x + t_x, s_y b_y + t_y))\| \\ &= \max_{b \in B} d'(s_x b_x + t_x, s_y b_y + t_y) \ . \end{aligned}$$

That is, $h(t(B), A)$ is simply the maximum of certain values from the Voronoi surface $d'(x)$ of the image set A. We can view this computation as placing $t(B)$ "on top of" the Voronoi surface of A and "probing" the surface at the places where the points of $t(B)$ lie. The largest of these probe values will specify $h(t(B), A)$. The reverse distance $f_A(t) = h(A, t(B))$ is defined analogously, in terms of probing the distance transform of $t(B)$.

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. For simplicity, we first consider just the forward distance $f_B(t) = h(t(B), A)$. Recall that, in effect each point of $t(B)$ 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.

A natural definition of the "partial distance" for K of the q points of B $(1 \leq K \leq q)$ is thus given by taking the K-th ranked point of $t(B)$ (rather than the largest ranked one),

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

where $K^{\text{th}}{}_{x \in X}\, f(x)$ denotes the K-th ranked value of $f(x)$ over the set X. For example, the *n-th* ranked value is the maximum, and the $n/2$-*th* ranked value is the median (if X has n elements).

In other words, the distance from each point of $t(B)$ to the closest point of A is computed, and then the points are ranked by their respective values of this distance. The K-th ranked such value, d, tells us that K of the model points $t(B)$ are each within a distance d of some image point. 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. When $K = q$ all of the points are considered, and we simply have a different way of writing the forward distance $h(t(B), A)$).

In order to compute the partial forward distance $h_K(t(B), A)$, we generally specify some fraction $0 < f_1 \leq 1$ of the points of B that are to be considered and set $K = \lfloor f_1 q \rfloor$. That is, we seek the distance where some given fraction, f_1, of the model points $t(B)$ lie near image points. This fraction is set based on how much occlusion of the model we wish to tolerate (what fraction of the model can be missing from the image). Thus for example if $K = \lfloor .9q \rfloor$, then $h_K(t(B), A) = \delta$ when 90% of the points of $t(B)$ lie within distance δ of some point of A.

One key property of h_K is that it does not require one to pre-specify which part of the model is to be compared with the image. That is, the method *automatically selects* the K "best matching" points of $t(B)$ for the given transformation t, because it identifies the subset of the model of size K such that the forward distance is smallest. In Section 8 we illustrate this partial matching capability.

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

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

From now on we use the partial distances h_K, h_L, and H_{LK} because as noted above the partial distance is a more general concept (i.e., when $L = p$ and $K = q$ the partial distance is the Hausdorff distance, where recall that p and q are the sizes of A and B respectively).

4 A Multi-Resolution Approach to Computing the Distance

As discussed in Section 1.1, we wish to identify those transformations, t, such that the Hausdorff distance is small, as these are good "matches" of the model to the image. That is, we must determine for what values of t (for what portions of the transformation space) $f(t) = H_{LK}(A, B(t)) \leq \tau$ for some threshold τ. This can be viewed as a search problem in the four-dimensional space of translation and independent x, y scaling. In [8, 9] the approach was to search the transformation space by quantizing it into buckets of some fixed size. Here we examine how to rule out large portions of the space by quantizing it into buckets of different resolutions. We use properties of the Hausdorff distance to rule out large areas of the space.

We first need a result concerning how the value of $f(t)$ changes for neighboring values of t. In essence, it says that since the slope of the function $f(t) = H_{LK}(A, t(B))$ is linear, the value cannot change very quickly. Thus if $f(t)$ is large for some value of t it must also be relatively large for all nearby values t'.

Claim 1 *Suppose that bounds on the points of the model set B are $0 \leq b_x \leq x_{\max}$ and $0 \leq b_y \leq y_{\max}$ for all $b = (b_x, b_y) \in B$. Let $t = (t_x, t_y, s_x, s_y)$ be any transformation of B and let $\delta = H_{LK}(A, t(B))$. Let $t' = (t'_x, t'_y, s'_x, s'_y)$ be any other transformation, and let $\delta' = H_{LK}(A, t'(B))$. Define*

$$\rho(t, t') = \left\| \left(|s_x - s'_x| x_{\max} + |t_x - t'_x|, |s_y - s'_y| y_{\max} + |t_y - t'_y| \right) \right\| .$$

(Note that ρ depends on B). Then

$$|\delta' - \delta| \leq \rho(t, t') . \tag{6}$$

Proof. Let $b = (b_x, b_y)$ be any point of B. Then $\beta = (\beta_x, \beta_y) = (s_x b_x + t_x, s_y b_y + t_y)$ is the location of $t(b)$. Similarly, $\beta' = (\beta'_x, \beta'_y) = (s'_x b_x + t'_x, s'_y b_y + t'_y)$. First we note that the distance between β and β' is bounded by the difference between t and t',

$$\begin{aligned} \|\beta - \beta'\| &= \left\| \left((s_x - s'_x) b_x + (t_x - t'_x), (s_y - s'_y) b_y + (t_x - t'_x) \right) \right\| \\ &\leq \left\| \left(|s_x - s'_x| b_x + |t_x - t'_x|, |s_y - s'_y| b_y + |t_y - t'_y| \right) \right\| \\ &\leq \left\| \left(|s_x - s'_x| x_{\max} + |t_x - t'_x|, |s_y - s'_y| y_{\max} + |t_y - t'_y| \right) \right\| \end{aligned}$$

Let $\beta_1, \ldots, \beta_K$ be the K points of $t(B)$ which are within distance δ of some point in A: $d'(\beta_i) \leq \delta$ for $i = 1, \ldots, K$. These points must exist since $h_K(t(B), A) \leq \delta$. Let b_i be the point in b such that $\beta_i = t(b_i)$, and let $\beta_i' = t'(b_i)$. Then for $i = 1, \dots, K$, we have

$$d'(\beta_i') \leq d'(\beta_i) + \|\beta_i - \beta_i'\| \leq \delta + \rho(t, t')$$

since $d'(\beta_i) = \|\beta_i - a\|$ for some $a \in A$, and so $d'(\beta_i') \leq \|\beta_i' - a\|$ (as $d'(\beta_i')$ is the distance from β_i' to the closest point of A). Thus, there are K points in $t'(B)$ which lie within $\delta + \rho(t, t')$ of some point in A, and so $h_K(t'(B), A) \leq \delta + \rho(t, t')$.

An analogous argument shows that $h_L(A, t'(B)) \leq \delta + \rho(t, t')$, since if $t(b)$ is the nearest neighbor in $t(B)$ of some point $a \in A$, then $t'(b)$ cannot have moved more than $\rho(t, t')$ farther away. Thus, $\delta' = H_{LK}(A, t'(B)) \leq \delta + \rho(t, t')$, and by symmetry the result follows. ∎

This result tells us that if $H_{LK}(A, t(B))$ is large for some transformation t, then it will also be large for any "nearby" transformation t'. Thus, if $H_{LK}(A, t(B)) = v > \tau$, then any transformation t' with $\rho(t, t') \leq v - \tau$ cannot have $H_{LK}(A, t'(B)) \leq \tau$. We use this fact to search efficiently for regions in the transformation space where H_{LK} is no greater than τ.

Let $R = [t_x^{\text{low}}, t_x^{\text{high}}] \times [t_y^{\text{low}}, t_y^{\text{high}}] \times [s_x^{\text{low}}, s_x^{\text{high}}] \times [s_y^{\text{low}}, s_y^{\text{high}}]$ be a rectilinear region in the transformation space. Let

$$t_c = \left((t_x^{\text{low}} + t_x^{\text{high}})/2, (t_y^{\text{low}} + t_y^{\text{high}})/2, (s_x^{\text{low}} + s_x^{\text{high}})/2, (s_y^{\text{low}} + s_y^{\text{high}})/2\right)$$

be the center of this region. Let $\delta = H_{LK}(A, t_c(B))$. If

$$\delta > \tau + \left\| \left(x_{\max}(s_x^{\text{high}} - s_x^{\text{low}})/2 + (t_x^{\text{high}} - t_x^{\text{low}}), y_{\max}(s_y^{\text{high}} - s_y^{\text{low}})/2 + (t_y^{\text{high}} - t_y^{\text{low}})\right) \right\| \quad (7)$$

then no transformation $t' \in R$ can have $H_{LK}(A, t'(b)) \leq \tau$. Conversely, if δ is smaller than this value, then it is possible that there is some transformation $t' \in R$ with $H_{LK}(A, t'(b)) \leq \tau$. This leads us to a multi-resolution method for searching the space of transformations:

1. Start with some rectilinear region ("cell") in transformation space which is known to contain all transformations of interest. Initialize a list of *possibly interesting* cells to this single cell.

2. Scan over the current list of cells. For each cell, determine the value of $H_{LK}(A, t_c(B))$ at its centerpoint t_c, and determine using equation (7) if it is possible that the cell contains a transformation t for which $H_{LK}(A, t(B)) \leq \tau$ If so, mark this cell as *interesting.*

3. Once the entire list has been scanned, make up a new list of smaller cells (all the same size) that completely cover those cells found to be interesting. Repeat steps 2 and 3 with this new list of (smaller) possibly interesting cells. Terminate when the current

cell size becomes small enough (this will be made more formal below).

Initially, the cells will be very large and almost every cell will be labeled as "interesting", since the interest threshold is dependent on the cell size. However, those cells which actually do not contain any interesting regions will be eliminated as they are subdivided and the subdivisions found not to be interesting. The final result will be a list of regions which represent transformations of the model which bring it into very close correspondence with (a section of) the image.

5 The 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 $A = \{a_1, \dots, a_p\}$ and $B = \{b_1, \dots, b_q\}$ 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 $B[k,l]$.

We primarily consider the computation of the forward distance $h_K(t(B), A)$, because as noted above the reverse distance is only computed to "verify" those transformations for which the forward distance is small. We compute the distance transform (cf. [5]) of the image set A, which we denote by $D'[x,y]$. This array specifies for each pixel location (x,y) the distance to the nearest nonzero pixel of A. 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 this distance transform array (e.g., [3, 10]).

5.1 Rasterizing Transformation Space

We must now determine how to rasterize the four-dimensional space of transformations: how to determine some grid in transformation space such that the transformations lying on the grid are somehow representative. If t and t' are two transformations which are adjacent in the rasterized transformation space (i.e. neighbors on the grid of transformations), we want the points of $t(B)$ to be adjacent (on the image grid) to the points of $t'(B)$. This leads us to the following rules:

1. The translational component of the translation should be rasterized to an accuracy of one pixel: translations should have integer components.

2. If for each point $b = (b_x, b_y) \in B$, we have $0 \leq b_x \leq x_{\max}$ and $0 \leq b_y \leq y_{\max}$ for any point, then we should rasterize the x-scale component to an accuracy of $1/x_{\max}$ and the y-scale component to an accuracy of $1/y_{\max}$. Thus, if $t = (t_x, t_y, s_x, s_y)$ and $t' = (t_x, t_y, s_x, s'_y)$ are adjacent in the grid of transformations, $s'_y = s_y \pm 1/y_{\max}$ and so for any $b \in B$, the y-coordinate of $t'(b)$ will be at most $b_y/y_{\max} \leq 1$ away from the y-coordinate of $t(b)$. Our rule is therefore that the x-scale component should be an integer multiple of $1/x_{\max}$, and the y-scale component should likewise be an integer multiple of $1/y_{\max}$. Note that this rasterization depends on the model B.

Transformations can therefore be represented by four integers; the quadruple (i_x, i_y, j_x, j_y) would represent the transformation $(i_x, i_y, j_x/x_{\max}, j_y/y_{\max})$.

It is also necessary to bound the range of the space of transformations which will be searched. We assume that the model is given at "full scale", and so we search only for transformations for which s_x and s_y are between 0 and 1; this corresponds to j_x values of between 0 and $x_{\max}$, and j_y values of between 0 and $y_{\max}$. We also restrict the range of translations: the scaled model must lie entirely inside the image array bounds.

Very small scale values can lead to problems: the entire model may be shrunk so much that it matches at every nonzero pixel in the image. We therefore put lower limits, $s_x^{\min}$ and $s_y^{\min}$, on s_x and s_y. If s_x is much greater than s_y, or vice versa, the transformed model will have a grossly distorted aspect ration. To allow this effect to be controlled, we have a parameter, $\alpha_{\max}$; any transformation for which $s_x/s_y > \alpha_{\max}$ or $s_y/s_x > \alpha_{\max}$ is ruled to be outside the space of transformations under consideration (the "valid transformations").

Suppose that the array $A[k, l]$ representing the set A has bounds $0 \leq k \leq m_a$ and $0 \leq l \leq n_a$. These restrictions correspond to the constraints

$$\begin{array}{rcl} s_x^{\min} x_{\max} & \le j_x \le & x_{\max} \\ s_y^{\min} y_{\max} & \le j_y \le & y_{\max} \\ \dfrac{y_{\max}}{x_{\max}\alpha_{\max}} & \le \dfrac{j_x}{j_y} \le & \dfrac{\alpha_{\max} y_{\max}}{x_{\max}} \\ 0 & \le i_x \le & m_a - j_x \\ 0 & \le i_y \le & n_a - j_y \end{array} \tag{8}$$

Proceeding with the analogy to the continuous case, we can compute the forward distance by probing locations of the image distance transform array, $D'[x, y]$, specified by the coordinates of the transformed model. We are limited by the rasterization accuracy of the integer grid, and so we must round each of the coordinates of the transformed model to an integer in order to probe $D'[x, y]$. The rasterized approximation to the forward distance, $h_K(t(B), A)$, is thus given by

$$F_B[i_x, i_y, j_x, j_y] = \underset{b \in B}{K^{\text{th}}} D'[\langle j_x b_x / x_{\max} + i_x \rangle, \langle j_y b_y / y_{\max} + i_y \rangle] . \tag{9}$$

where $\langle \cdot \rangle$ denotes rounding to the nearest integer.

In order for F_B to be small at some transformation (i_x, i_y, j_x, j_y), it must be that the distance transform $D'[x, y]$ is small at the K of the q locations $(\langle j_x b_x / x_{\max} + i_x \rangle, \langle j_y b_y / y_{\max} + i_y \rangle)$ specified by the points $(b_x, b_y) \in B$. In other words, K of the q points (nonzero pixels) of the transformed model must be near some point (nonzero pixel) of the image.

The rounding of the points of $t(B)$ required by this rasterization introduces only a bounded (constant) error in F_B, since the points move only slightly and the distance transform array may not change quickly (its slope is of magnitude at most 1); similarly, any transformation (in the allowable range of transformations) can be approximated by some transformation on the raster grid with only a small error introduced: if t is approximated by its nearest neighbor $t' = (i'_x, i'_y, j'_x / x_{\max}, j'_y / y_{\max})$ on the grid, then $F_B[i'_x, i'_y, j'_x, j'_y]$ differs from $h_K(t(B), A)$ by only a few pixels; the exact number depends on the norm used to compute the distance transform array (generally L_1, L_2, or L_∞).

We may also approximate the reverse distance, $h_L(A, t(B))$ using similar methods. Here, we must first round the points of $t(B)$ to integer coordinates and compute their distance transform. We then "probe" this array at the locations of the points of A. The L-th ranked probe value is then a close approximation to $h_L(A, t(B))$, which we will refer to as $F_A[i_x, i_y, j_x, j_y]$. The bidirectional distance is simply

$$F[i_x, i_y, j_x, j_y] = \max(F_A[i_x, i_y, j_x, j_y], F_B[i_x, i_y, j_x, j_y]) \tag{10}$$

where we use $K = \lfloor f_1 q \rfloor$ of the model points and $L = \lfloor f_2 p \rfloor$ of the image points for the partial distance ($0 < f_1, f_2 \leq 1$). Recall that in computing F we first compute the forward distance F_B, and only compute the reverse distance F_A for those transformations where F_B is small.

5.2 Reverse Distances in Cluttered Images

In many applications the model is considerably smaller than the image, reflecting the fact that a single instance of the model will occupy only a small portion of the image. In such cases, the above definition of the reverse distance is not ideal, because we must somehow determine an appropriate value of L, the number of image pixels that will be close to model pixels. This number, L, however will depend on what other objects are in the image.

A natural way to handle this is to compute a partial reverse distance only for 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. Thus we compute a different version of $F_A[i_x, i_y, j_x, j_y]$ that considers just the points of $A[k, l]$ that are under the model at its given position:

$$F_A[i_x, i_y, j_x, j_y] = \max_{\substack{(k,l) \\ i_x \leq k < i_x + j_x \\ i_y \leq l < i_y + j_y}} A[k, l] D[k, l], \tag{11}$$

where $D[k, l]$ is the distance transform of $t(B)$.

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 in equation (11) would be modified to use L^{th} instead of max. This is the definition that we use in practice.

6 Rasterizing the Multi-Resolution Method

We now turn to the rasterized version of the multi-resolution approach introduced in Section 4. As described above, we only consider transformations which lie on a grid in transformation space. The pitch of this grid is 1 in each of the translation dimensions, $1/x_{\mathrm{max}}$ in the x-scale dimension, and $1/y_{\mathrm{max}}$ in the y-scale dimension. The algorithm attempts to find all transformations within the space of valid transformations (as described in equation (8)) for which $F_B[i_x, i_y, j_x, j_x] \leq \tau$, for some specified threshold τ. Then for those transformations where $F_B \leq \tau$, the reverse distance F_A is computed in order to find the transformations where both distances are small.

Our algorithm begins with a single grid cell which encompasses the entire space of valid transformations. It proceeds through several levels of increasingly fine resolution. At each level, it begins with a list of cells which are *possibly interesting.* These cells are all of the same size. It then determines, for each cell, if it is possible that it contains some transformation whose forward distance, $F_B[i_x, i_y, j_x, j_x]$, is less than or equal to τ. This is done by computing F_B for a single transformation at (or near) the center of the cell, and using equation (7) to set a threshold which determines whether or not it is possible that the cell contains some transformation with a low F_B value. Note that this threshold depends only on the size of the cell, and not on its location; it will therefore be the same for all cells at the current level. Call this threshold τ'.

We consider the transformation which is at the center of the cell (or the closest grid transformation to the center, in the case that the center does not lie on the grid), and compute its F_B value. If this is greater than τ', then the cell is labeled *uninteresting* and removed from consideration. If it is less than or equal to τ', then the cell is labeled *interesting.* Once all the cells at the current level have been tested, the list of possibly interesting cells for the next level is created. This is done by taking all those cells which were labeled interesting and determining a set of finer-resolution cells which completely cover them. These finer-resolution cells may not be subdivisions of the coarser-resolution cells: one finer-resolution cell may overlap more than one coarser-resolution cell. This is done so that the finer-resolution cells can all have the same size without overlapping each other. (Note that simply subdividing the coarser-resolution cells would not achieve this, because the cells must have integer lengths.) Having all the cells be the same size allows us to use some heuristics to accelerate the computation; these are discussed in Section 7.

The finer-resolution cell size is found by taking the size of the coarser-resolution cell and dividing each dimension by a fixed constant R (we have been using $R = 2$), and rounding each to an integer number of grid steps. Any dimension which would be smaller than a single grid step is held at one grid step.

If the cells at the current level are a single grid step wide in each dimension (i.e. each cell includes only one transformation), then no further refinement is possible. At this point, the list of interesting cells from the current level contains all those transformations for which $F_B[i_x, i_y, j_x, j_y] \leq \tau$. We then consider each of these individually. For each one, we compute the distance transform of the transformed model, and use this to compute $F_A[i_x, i_y, j_x, j_y]$ using equation (11). We reject all those transformations for which this is greater than τ. Thus, we are left with a list of all the transformations for which $F[i_x, i_y, j_x, j_y] \leq \tau$.

7 Increasing the Efficiency of the Computation

There are several techniques which we have found to be useful in making this multi-resolution search more efficient. They involve determining when it is possible to say that a cell is "interesting" or "uninteresting" without completely calculating F_B at its center. The multi-resolution search described in the previous section only uses the value of F_B at the center of each cell to test if it is smaller than or greater than τ' (where τ' is the threshold that depends on the cell size). This means that if we can make the comparison with τ' efficiently, without computing the actual value of F_B, we may achieve improved efficiency.

7.1 Early Rejection

The process of computing $F_B[i_x, i_y, j_x, j_y]$ as in equation (9) involves probing $D'[x, y]$ at the locations determined by the transformed points of B, and taking the K-th of these values (where $K = \lfloor f_1 q \rfloor$). We therefore keep count of the number of probe values seen which are greater than τ'; if this count ever exceeds $q - K$, then we need probe no more locations for this cell. This method works best for large values of f_1 (and is easiest to understand when $f_1 = 1$, in which case a single probe value greater than τ' rules out the cell).

7.2 Early Acceptance

Consider the consequences of an error in the classification of a cell as interesting or uninteresting. If we mistakenly classify a cell as uninteresting, the entire volume covered by that cell will be eliminated from any further consideration. Our search will thus miss any transformations in that cell that bring B to within τ of A. We therefore cannot allow this form of error.

On the other hand, if we mistakenly classify a cell as interesting, when it actually is not, then all that we have done is added a few more finer-resolution cells to the list of possibly interesting cells at the next level. These cells will be probed and eliminated themselves; our search will not turn up any "false positives", unless this mistake is made at the finest resolution.

Thus, if we mistakenly classify a cell (not at the finest resolution) as interesting when it is not, we have not actually committed an error, and the search will succeed, though it will have done extra work. However, this allows us to use a heuristic method to make the decision as to whether a cell is interesting or not, as long as it only makes "false positive" errors, and no "false negative" errors. The overall time spent by the search may be reduced if this heuristic is less expensive to compute than a full computation of F_B.

The heuristic that we are using randomly selects some fraction s, $0 \le s \le 1$, of the points of B. It probes the distance transform of A at the transformed locations of these selected points. If the fraction of the probe values which are less than or equal to τ' is more than f_1, then we say that this cell is interesting, and do not perform any further probes. If the fraction is less than f_1, then we perform the rest of the probes and make our decision based on the full set. In fact, we may not have to probe at the transformed locations of all of the points of B: as in the previous subsection, we can terminate the scan (and rule the cell uninteresting) if more than $q - K$ of the probes so far are greater than τ'.

At present, we are using $s = .2$: we perform the first 20% of the probes before making any decision. In the examples we have looked at, this causes a false positive rate of about .1. That is, about 10% of the cells labeled as interesting actually could have been shown to be uninteresting. However, the time saved by not probing every point is much greater than the time spent correcting these errors later. It is important that the points used be chosen randomly from B; if they are all clustered in one part of B, then the false positive fraction rises greatly.

Of course, we do not use this heuristic at the finest resolution. At this resolution we need to actually compute F_B for all the transformations that are still possibly valid, since this (along with F_A) determines which transformations are actually valid (have $F \le \tau$)

7.3 Skipping Forward

A third technique relies on the order in which the list of possibly interesting cells is scanned. We must be scanning this list in some order; assume that the scan considers cells which are adjacent in the x-scale dimension in sequence, in the direction of increasing scale. In other words, for some i_x, i_y and j_y, we consider the cell having center with coordinates $(i_x, i_y, 0, j_y)$, then $(i_x, i_y, 1, j_y)$, and so on (of course, cells that are not in the list are not considered). It may therefore be possible to rule out entire stretches of this scan: at (i_x, i_y, j_x, j_y) we may be able to compute a value Γ_x such that all cells from (i_x, i_y, j_x, j_y) to $(i_x, i_y, j_x + \Gamma_x - 1, j_y)$ can immediately be ruled uninteresting. To do this, we use a variant of the distance transform of A.

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, a value greater than the width of the array is used). Formally,

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

Note that $D'_{+x}[x, y] \geq D'[x, y] - \tau'$, and $D'_{+x}[x, y] = 0$ exactly when $D'[x, y] \leq \tau'$. Computing $D'_{+x}[x, y]$ is straightforward once $D'[x, y]$ is known: we simply scan right-to-left along each row of $D'[x, y]$, keeping track of the distance to the nearest location whose value is under τ'. Since τ' changes at every level of our multi-resolution search, we must recompute $D'_{+x}[x, y]$ for each level. (Note that if the cells at each level were not all the same size, then we might have to recompute this for every cell, which would more than negate the time savings.)

We can use $D'_{+x}[x, y]$ to determine how far we would have to move in the increasing x-scale direction to find a place where $F_B[i_x, i_y, j_x, j_y]$ might be no greater than τ'. Consider a point $b = (b_x, b_y) \in B$. Let

$$\Delta_x = D'_{+x}[\langle j_x b_x / x_{\max} + i_x \rangle, \langle j_y b_y / y_{\max} + i_y \rangle] \, .$$

Now, we want to determine how much we might have to increase j_x before b's transformed position falls on a point of $D'[x, y]$ whose value is under τ'. Each increase in j_x moves b's transformed position by $b_x / x_{\max}$. Its position must move Δ_x pixels in $D'[x, y]$ before $D'[x, y]$'s value can be under τ'. We must allow for the fact that the transformed position is rounded before the probe into $D'[x, y]$ is made. Thus, a conservative estimate of this necessary increment is

$$\gamma_x = \frac{x_{max}}{b_x} \max(0, D'_{+x}[\langle j_x b_x / x_{\max} + i_x \rangle, \langle j_y b_y / y_{\max} + i_y \rangle] - 1) \; .$$

Now, $F_B[i_x, i_y, j_x, j_y], \ldots, F_B[i_x, i_y, j_x + \gamma_x - 1, j_y]$ must all be greater than τ'.

Let

$$\Gamma_x = \underset{b \in B}{K^{\text{th}}} \frac{x_{\max}}{b_x} \max(0, D'_{+x}[\langle j_x b_x / x_{\max} + i_x \rangle, \langle j_y b_y / y_{\max} + i_y \rangle] - 1)$$

Γ_x thus represents a conservative estimate of how far away in the increasing x-scale dimension the nearest grid point is for which it might be that $F_B \leq \tau'$. Any cell whose center is closer than this can immediately be ruled uninteresting.

Note that the probed values of $D'_{+x}[x, y]$ also provide exactly the information we need to apply the previous two acceleration techniques: both relied only on counting how many of the $D'[x, y]$ values that they probed were greater than or less than τ'. This is equivalent to counting how many of the $D'_{+x}[x, y]$ values probed are nonzero. We therefore do not need to probe $D'[x, y]$ at all during the search, except at the final level, when $F_B[i_x, i_y, j_x, j_y]$ must be explicitly calculated.

This technique of skipping forward interacts with the technique of early rejection described in Subsection 7.1: the cells which are subject to early rejection are likely to be exactly those whose Γ_x values would be large, enabling the elimination of other cells. If we have only probed at some of the points of the transformed model, we can obtain a conservative value for Γ_x by assuming that all the unprobed values would have been zero; this allows us to perform some skipping forward. We can also modify the early rejection technique by continuing to probe until Γ_x can be shown to be at least $\Gamma_{\min}$, for some value of $\Gamma_{\min}$. This strengthens the skipping-forward technique, but weakens the effect of the early rejection technique.

The skipping-forward technique is less useful at coarser resolutions, since the distance between cell centers is greater, and so a larger Γ_x is required before any cells other than the current one can be eliminated. $\Gamma_{\min}$ could therefore be adjusted as the resolution becomes finer. We have not experimented with this idea.

8 Examples

The group of two-dimensional translations and scaling in x and y is useful for certain problems involving two-dimensional images of objects in the three-dimensional world. An object which translates in three-dimensional space will appear to translate and scale uniformly in

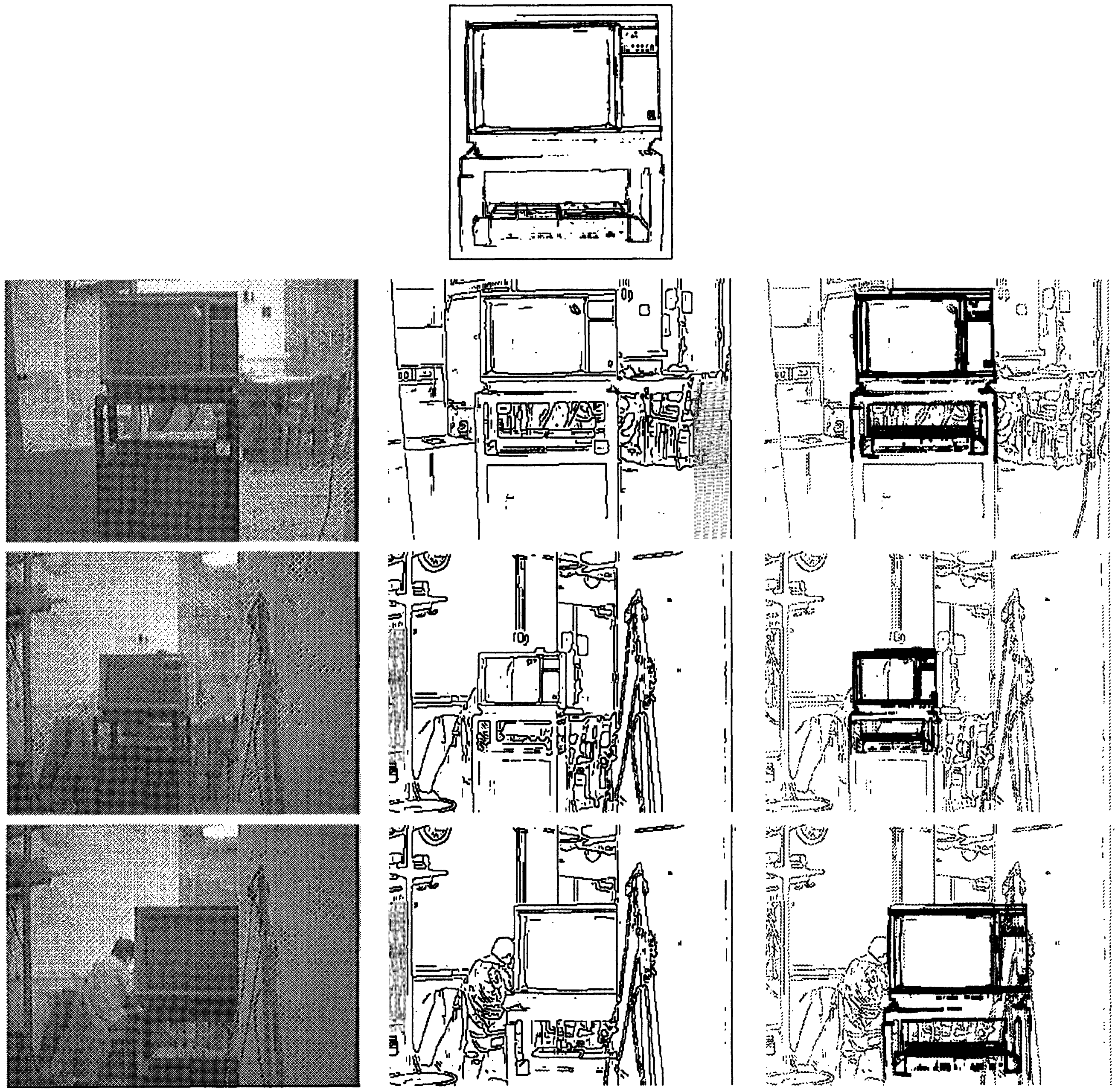

Figure 1: The model at full scale, and the three test images for the first example.

the image. If it rotates slightly either about a vertical axis, or about a horizontal axis which is perpendicular to the camera axis, this will appear as a scaling in x or y, respectively. These two effects can be produced by the camera base panning or tilting, as it might if mounted on a pan-tilt head.

We now consider some examples in order to illustrate the performance of the Hausdorff distance methods developed above. The first test model is shown in the top row of Figure 1.

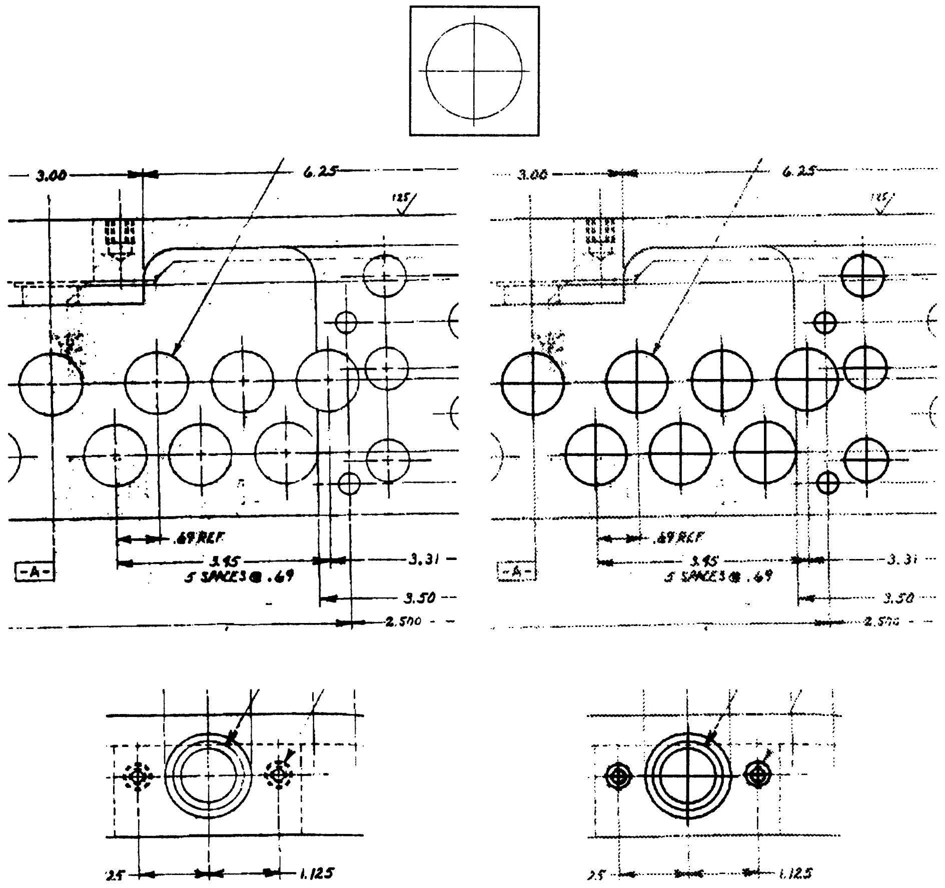
Figure 2: Finding circles in an engineering drawing

This binary image is 190 $\times$ 220 pixels, and was produced by applying an edge operator 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 timings given are user-mode CPU times for a SPARCstation-2.

We compared this model against three test images. Each image is a half-size NTSC video frame (320 $\times$ 240 pixels). For each test image, we show in Figure 1 the gray-level image, its edges, and the edges overlaid with the scaled and translated model.

For the first two test images, we used $\tau = 2\sqrt{2}$ pixels, $f_1 = 0.95$, $f_2 = 0.75$, $s_x^{\min} = s_y^{\min} = 0.4$ and $\alpha_{\max} = 1.02$. In the third test image, however, the model object is partially out of view. In order to allow it to be matched, we reduced f_1 to 0.85, allowing up to 15% of the model points to match nothing in the image. We also used $\tau = 2\sqrt{2}$, $f_2 = .65$, $s_x^{\min} = s_y^{\min} = 0.7$ and $\alpha_{\max} = 1.02$. Running times for these three examples was between 90 and 250 seconds each.

Each example generated multiple matches; in each case, these formed a single connected component in the transformation space (i.e. were essentially the same transformation). For each test image, we therefore picked the best match as a representative; this is shown overlaid on the image.

Our second example involves finding circles in an engineering drawing of a part to be machined; this might be done in order to automatically determine where holes should be drilled. The model is shown in the top row of Figure 2. It is 175×175 pixels, and contains a circle of radius 75 pixels.

Since engineering drawings typically contain text, arrows, and overlapping parts, the reverse distance is not a reliable indication of the quality of the match: if the model matches some circle which happens to be crossed by a line, then the reverse distance will be increased, even though the match might otherwise be good. For this reason, we have used only the forward distance, $F_B[i_x, i_y, j_x, j_y]$, as an indicator of match quality. The results of these tests are shown in Figure 2. We show the original bitmap of the two test images, then the original bitmaps overlaid by the scaled and translated model. The bitmap is from a blueprint which was photoreduced and scanned at 150dpi. Note that some of the circles have been broken up, and there is some background noise due to the speckling of the original blueprint.

The first example shows several circles of different sizes. Using $\tau = 1$, $f_1 = 0.9$, $s_x^{\min} = s_y^{\min} = 0.2$ and $\alpha_{\max} = 1.01$, our method correctly located all the circles. The matches which our method detected formed multiple connected components in transformation space, one component for each circle in the image. We display the best match for each component.

The second example illustrates how we can find circles drawn with dotted lines, using the same (solid) circle model. With $\tau = 1$, $f_1 = 0.82$, $s_x^{\min} = s_y^{\min} = 0.14$ and $\alpha_{\max} = 1.01$, again all the circles were correctly located. Due to the low resolution of the scan, the dotted circles were too close to be reliably separated. Running times for these two examples were between 200 and 250 seconds.

9 Summary

We have presented a method for efficiently computing the Hausdorff distance between a model and an image, where the model is allowed to translate and scale (separately in x and y). This method improves considerably on techniques presented in previous papers. The main advance is the development of a multi-resolution method for searching the space of possible transformations of a model with respect to an image. This allows large portions of the space to be ruled out relatively cheaply. This multi-resolution method is augmented with a random sampling strategy, which is able to quickly estimate whether or not a given region of the transformation space should be ruled out. This random sampling strategy is guaranteed not to miss a match.

The method was illustrated using examples from two problem domains. The first domain is matching intensity edges extracted from grey-level images of indoor scenes. This is useful in problems where two-dimensional images are taken of an object that is translating (but not rotating) in the three-dimensional world. The second domain is the interpretation of engineering drawings, where tokens are generally at an unknown location and scale, but at certain canonical orientations.

References

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

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

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

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

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

[6] 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.

[7] 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. To appear in *Discrete Computational Geometry.*

[8] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pat. Anal. and Mach. Intel.* To appear.

[9] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the Hausdorff distance under translation. In *Computer Vision and Pattern Recognition*, pages 654–656, Champaign-Urbana, Illinois, 1992.

[10] 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.

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