Line-Based Recognition Using A Multidimensional Hausdorff Distance

Xilin Yi, *Member*, *IEEE*, and Octavia I. Camps, *Member*, *IEEE*

Abstract—In this paper, a line-feature-based approach for model based recognition using a four-dimensional Hausdorff distance is proposed. This new approach reduces the problem of finding the rotation, scaling, and translation transformations between a model and an image to the problem of finding a single translation minimizing the Hausdorff distance between two sets of points in a four-dimensional space. The implementation of the proposed algorithm can be naturally extended to higher dimensional spaces to efficiently find correspondences between n-dimensional patterns. The method performance and sensitivity to segmentation problems are quantitatively characterized using an experimental protocol with simulated data. It is shown that the algorithm performs well, is robust to occlusion and outliers, and that it degrades nicely as the segmentation problems increase. Experiments with real images are also presented.

Index Terms—Hausdorff distance, line-feature-based recognition, multidimensional distance transform.

1 Introduction

The Hausdorff distance measures the distances between two sets of points. Thus, it can be used to measure the similarity between two patterns of points when they are superimposed on one another. A correlational method using the Hausdorff distance to determine if there is any model pattern in a given image was proposed by Huttenlocher et al. [8], [7], and extended by Rucklidge [14], [16]. This method can be used in a wide range of applications, such as real scene recognition, tracking [14], engineering drawing understanding, and aerial image analysis [19].

Two important properties of this approach are 1) the use of the Hausdorff distance makes it robust to data uncertainty and the presence of outliers and 2) implementations of the algorithm using point features (edge pixels or *edgels*) have been developed, some of them exploiting specialized computer graphics hardware [8]. While this approach works well and it is computationally efficient in the presence of model translation in the image, it is significantly time consuming when the model has also been rotated and scaled [14].

In this paper, a line-feature-based four-dimensional recognition algorithm that overcomes the above problem is proposed. The main idea of the new approach is to reduce the problem of finding translation, rotation, and scaling transformations in the (two-dimensional) image domain to the problem of finding a translation transformation in a four-dimensional space. This is accomplished by using line segments as features represented in a four-dimensional space defined as follows. Each model and image segment is associated the coordinates of its mid-point (x, y), the logarithm of its length $\log l$, and its orientation θ, and is represented as a point in the $(x, y, \log l, \theta)$ space. Then, translations in the x, y directions, rotation, and scaling in the two dimensional image domain correspond to translations in the x, y, θ, and $\log l$ directions in the new four-dimensional domain, respectively.

In [18], we presented a matching algorithm using this approach that found a *suboptimal* solution by decomposing the problem of finding the four-dimensional translation into the problem of finding two two-dimensional translations, one corresponding to the rotation and scaling, the other corresponding to the translation in the image. In this paper, we present a new algorithm that finds the *optimal* solution, where the translation, rotation, and scaling between the model and the image are all simultaneously determined. As a result, the new algorithm overcomes many of the false alarms occurring with our previous solution.

We will first review the definition of the Hausdorff distance between two point sets and briefly summarize the algorithm proposed by Huttenlocher et al. in [8]. Next, we describe the new approach and how to find the translation minimizing a four-dimensional Hausdorff distance efficiently. It is shown that the running time of the proposed algorithm grows linearly with the dimension of the problem and that its memory requirements are significantly reduced by exploiting the data sparsity. Thus, the algorithm can be easily adapted to perform efficient pattern matching in higher dimensions [4]. Following that, we present an extensive system evaluation protocol using simulated data to characterize the performance of the algorithm in the presence of noise and segmentation problems. Finally, we present the test results with real images.

- *X. Yi is with ENSCO Inc., Springfield, VA 22151.*
- *O. Camps is with theDepartment of Computer Science and Engineering, The Pennsylvania State Unversity, University Park, PA 16802. E-mail: camps@whale.ee.psu.edu.*

Manuscript received 16 Dec. 1998; revised 9 Mar. 1999.
Recommended for acceptance by P. Flynn.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107755.

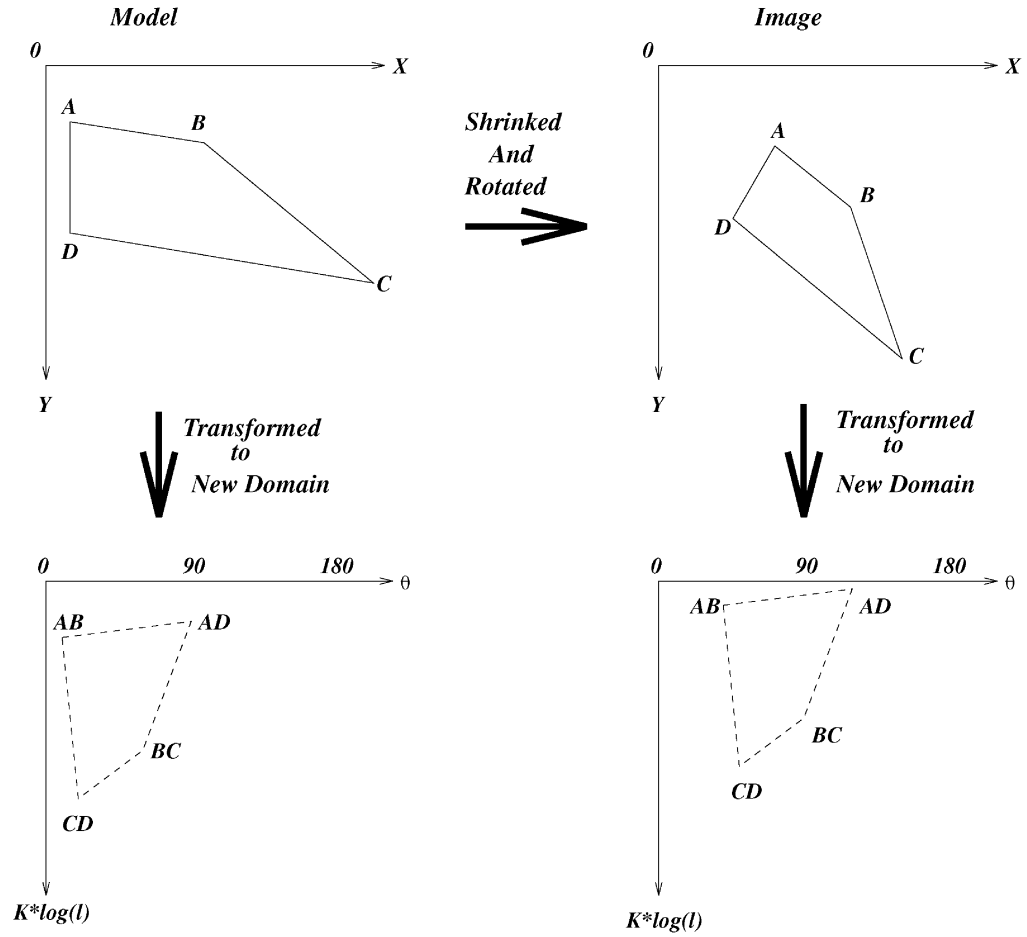


Fig. 1 Illustration of transforming the image and model to the $(\log l, \theta)$ domain.

2 The Hausdorff Distance Between Two Point Sets

Given two point sets A and B, the Hausdorff distance between A and B is defined as

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

where $h(A,B) = \max_{a\in A} \min_{b\in B} \|a-b\|$, and $\|.\|$ denotes some norm on the points of A and B. The functions $h(A,B)$ and $h(B,A)$ are called the *directed* Hausdorff distances from A to B and B to A, respectively.

Outliers can be handled by generalizing the directed distances $h(B,A)$ and $h(B,A)$ to the *partial* directed distances $h_K(B,A)$ and $h_L(A,B)$ [8]:

$$h_K(B,A) = K^{th}_{b\in B} \min_{a\in A} \|a-b\| \tag{2}$$

with $h_L(B,A)$ defined similarly. The partial bidirectional Hausdorff distance is then defined as

$$H_{LK}(A,t(B)) = max(h_L(A,t(B)), h_K(t(B),A)), \tag{3}$$

where $t(.)$ denotes a transformation of a set. The partial distance measures the difference between a portion of the model and the image and, thus, it is more robust to occlusion and outliers.

Dubuisson and Jain [6] have studied 24 different variations of this distance, including the one given above, and they have compared them in the presence of noise. Although their results showed that a modified Hausdorff distance (MHD) using the average distance between the points of one set to the other set gives the best result for object matching, we have chosen to use the partial Hausdorff distance defined above, which performs closely to the MHD, but requires substantially less memory with our algorithm.

If a transformation $t(.)$ is applied to the model point set, we are interested in the Hausdorff distance as a function of the transformation of the set B, or

$$d(t) = H_{LK}(A, t(B)). \tag{4}$$

Hence, to recognize a model-like image, we can apply different transformations to B: If $d(t)$ is less than a certain threshold, a match between the model and image can be hypothesized.

If A is a set of image points and B is a set of model points, the matching process of finding $d(t) = H_{LK}(A,t(B))$ is called forward matching. If a certain Hausdorff distance is set as a threshold to this process, a fraction of model points satisfying this threshold for a transformation $t(.)$ may be found. This fraction $f(t)$ is called the forward matching fraction. On the other hand, if A is translated instead of B,

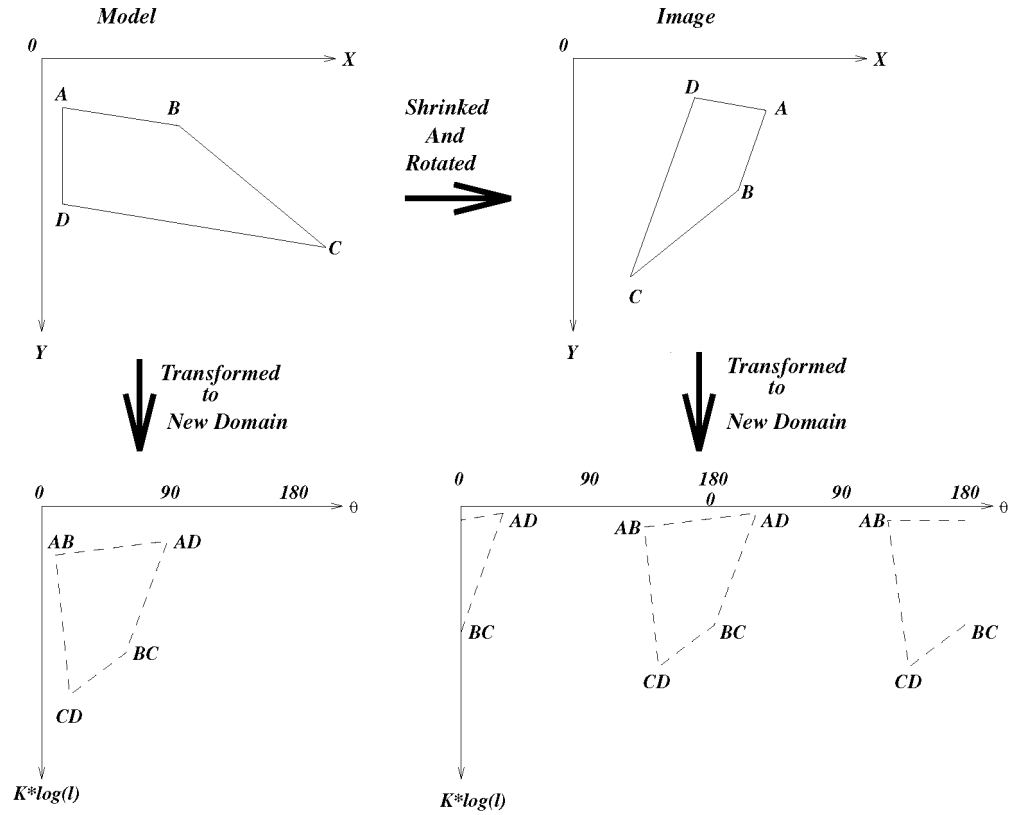


Fig. 2. Illustration of the modified version of transforming the image and model to the new domain.

this is called the reverse matching process. Accordingly, there are a reverse Hausdorff distance threshold and a reverse matching fraction $g(t)$.

3 Point Feature-Based Matching Algorithm

Huttenlocher et al. [8] developed an algorithm to search for the transformation $t(.)$ that minimizes the Hausdorff distance between two set of 2D points, $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$.

The main idea behind their algorithm is summarized in the following steps:

1. Compute the distance transform—i.e., the distance for any point (x, y) to the nearest point in a set of source points—for the sets A and B, using a two pass algorithm like the one in [1]. Let $D[x, y]$ denote the distance transform of A and $D'[x, y]$ the distance transform of B. That is, $D[x, y]$ and $D'[x, y]$ specify the distance between the pixel (x, y) and the nearest points in A and B, respectively.
2. Compute the pointwise maximum of all the translated distance transform arrays to determine the Hausdorff distance as a function of translation:

$$F[x, y] = \max\Big\{ \max_{a \in A} D[a_x - x, a_y - y], \max_{b \in B} D'[b_x + x, b_y + y] \Big\},$$

   where (a_x, a_y) and (b_x, b_y) are the 2D coordinates of $a \in A$ and $b \in B$.
3. Finally, find the translation that minimizes the function $F[x, y]$.

This algorithm also uses several pruning techniques—ruling out circles, early scan termination, and skipping forward—to avoid computing the Hausdorff distance for translations that cannot be a minimum, resulting in considerable time speed ups. The resulting algorithm is very efficient in the case that the transformation $t(.)$ is a pure translation. However, if $t(.)$ includes rotation and/or scaling, the algorithm requires computing the Hausdorff distance as a function of all possible transformations and, hence, the computational time increases dramatically [14], [15]. In this paper, we present a new algorithm, inspired by the one outlined above, that uses line features in a four-dimensional space to overcome this problem.

4 Proposed Approach

In this section, we introduce the use of line features to define a four-dimensional search space, the basic idea of our approach.

It can be observed that, when the model is rotated, the orientation angles of the line segments in the image are increased or decreased by a constant with respect to the

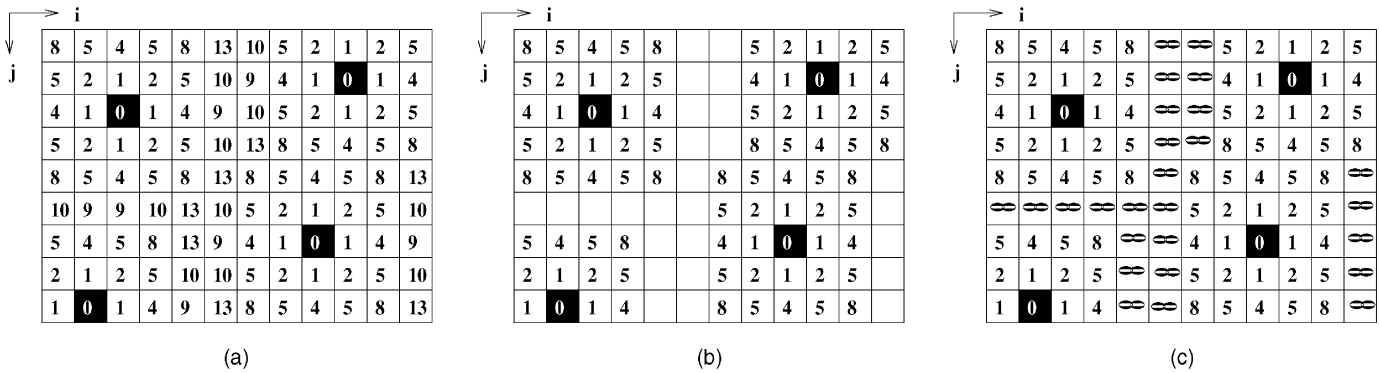


Fig. 3. Representation of distance transform of sparse data. (a) Distance transform for four sparse points, marked in black. (b) Distance transform for only those points with a distance transform less or equal to 8. (c) The distance transform for all other points is assumed to be infinity.

orientation of the corresponding model segments, but remain unaffected when the model is translated or scaled. On the other hand, the lengths of the image segments are multiplied by a constant factor when the model is scaled, but are invariant under translation and rotation. Next, we show how these properties can be used to find rotation and scaling transformations efficiently.

Model and image line segments can be represented by four-dimensional points with coordinates equal to their mid-point coordinates, the logarithm of their length, and their orientation. Note that, although the choice of using the logarithm of the length over using simply the length of the segment may be surprising at first glance, it is rather natural when the previous observations are taken into account.

Let s_m be a model segment with coordinates $(x_m, y_m, \log l_m, \theta_m)$. Consider an ideal image s_i of s_m obtained by rotating s_m by an angle θ_{Rot} and scaling it by a factor S, and the projections of the four dimensional points corresponding to s_m and s_i onto the plane $(\log l, \theta)$. The logarithm of the length and the orientation of the image segment s_i are given by

$$\log l_i = \log S + \log l_m$$
$$\theta_i = \theta_m + \theta_{Rot}.$$

Thus, it is seen that rotating and scaling a model segment is equivalent to translating the corresponding four-dimensional point along the directions $\log l$ and θ by an amount $\log S$ and θ_{Rot}, respectively.

To simply illustrate the above idea, consider the following example. Suppose, we have two polygons, where one is a rotated and scaled version of the other, as shown on the upper half of Fig. 1. Matching these polygons is equivalent to find the rotation and scaling transformations that transforms one into the other. The projections into the $(\log l, \theta)$ plane of the 4D representations of the given polygons are shown in the lower half of Fig. 1. These projections are the vertices of two polygons of identical shape and dimensions, differing only in a translation that corresponds precisely to the rotation and scaling factors in the original domain.

Without any further modifications, the image and model in the new domain will have a range in the θ axis from 0^o to 180^o. However, applying the translational Hausdorff distance to the projections of the transformed segments will not always result in the correct rotation angle θ_{Rot}. For example, if the model is rotated 110^o, the model segment AD that has an orientation of 90^o will have a corresponding image segment with an orientation of 20^o and the new polygons in the plane $(\log l, \theta)$ will not have the same shape. This problem can be easily solved by extending the image orientation range from 0^o to 360^o. The image pattern from 180^o to 360^o is an exact copy of the pattern from 0^o to 180^o. There is no need to modify the model pattern. Fig. 2 shows the modified approach. As we can see, in this modified version, the image pattern will be identical to the model pattern except for some spurious points in the lower and upper range of θ and, therefore, the image *scale* and *rotation* transformations can be found as a translation in the new $(\log l_m, \theta_m)$ domain. The remaining image *translation* transformation can be found as a translation in the new (x_m, y_m) domain, between the coordinates of the image midpoints and the rotated and scaled coordinates of the model midpoints.

However, before this approach can be used, there are two important issues that need to be addressed:

1. In the transformed domain, $\log l$ and θ are measured using different units. While applying Hausdorff matching, distances are computed. Thus, compatible measurements in both axes are needed. Adding a scaling factor to one of the axes is one way of solving this problem. In our approach, we use a K_s factor for the $\log l$ axis. This factor K_s is also related to the resolution of the $(\log l, \theta)$ plane in a discrete implementation. If a line has length uncertainty ρ, it is desirable for this line to fall into the same "bin" that the ideal line without uncertainty would. Thus, K_s must satisfy:

$$|K_s * log(l + l * \rho) - K_s * log(l)|$$
$$= |K_s * log(1 + \rho)| < 1$$

2. If a line is broken into two or more pieces, the Hausdorff distance using the $\log l$ and θ parameters will be incorrect. This is because the original line splits into lines with smaller lengths that have no

relation to the original length. Since this problem is not easy to fix, it is important to characterize the sensitivity of the algorithm to segmentation problems to evaluate its merit. This is done as part of our experimental protocol in Section 6.

5 Line Feature-Based Matching Algorithms

Once the model and image line segments are represented in the new 4D domain, the problem of finding the translation, rotation and scaling transformations that best align the model and image segments reduces to the problem of finding the best *translation* between the corresponding 4D points.

5.1 The 2D-2D Matching Algorithm

In [18], an algorithm was presented that attempted to solve this problem by decomposing it into the problem of finding two optimal 2D translations, one on the $(\log l, \theta)$ plane, the other on the image plane. Thus, this algorithm is referred to as the 2D-2D matching algorithm.

The first 2D translation provided the scaling factor and the rotation angle. Once these transformations were found, the model in the original image domain was rotated and scaled using these parameters. Then, the best translation between the updated model and the image was found by running the matching algorithm once more, this time on the image plane.

The main advantage of this approach is that the required 2D translations can be efficiently found by simply using twice the 2D point-based matching algorithm outlined in Section 3.

5.2 The 4D Matching Algorithm

There are some disadvantages with the 2D-2D matching algorithm outlined above. First, this procedure allows errors in the first matching step to propagate to the second matching step. For example, small errors in the rotation angle or the scaling factor may result in a distorted transformed model that will not be matched in the second step. Furthermore, if the rotation angle is between 180^o and 360^o, say $180^o + \phi$, the first matching step will also find ϕ as the rotation angle, since the length and orientation of the lines will match even if the image is rotated 180^o. Thus, this first matching will introduce false alarms that may or may not be eliminated by the second step and that will increase the running time of the algorithm in either case. Finally, finding the optimal translations in the two 2D domains does not guarantee that the obtained solution is optimal in the original 4D domain.

All of the above problems arise from the loss of some of the inherent geometrical constraints of the problem, caused by the artificial separation of the 4D data into two sets of 2D data. Hence, it is desirable to perform the search for the translation, rotation and scaling transformations simultaneously.

In principle, the algorithm summarized in Section 3 could be generalized to run in four dimensions in a straight forward manner. In practice, however, this is not possible due to the following considerations:

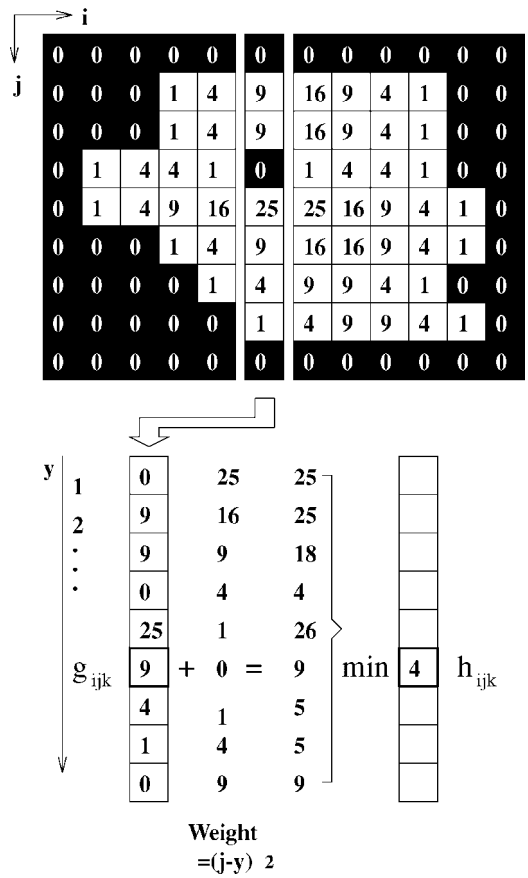


Fig. 4. Computation of the distance transform using the algorithm proposed by Saito and Toriwaki [17]. The black pixels correspond to foreground pixels. The image in the top shows the result of the first transformation for the kth slice of a three dimensional image. The bottom of the figure shows how to apply the second transformation to pixel $(6, 6, k)$.

Memory requirements. The large size of the transformed 4D domains makes, impractical to use arrays to store the patterns. Consider, for example, an image of size 480×512. Then, the corresponding 4D space should be 480×512 for the sliced x and y plane. The range for the rotation angle should be 180 for the model and 360 for the image, considering increments of one degree. The range for the logarithm of the length of the segments varies depending on the factor K_s. Since the longest possible line in the image is the diagonal, this range is, for a value of $K_s = 50$, equal to 142. Thus, the corresponding 4D spaces are $480 \times 512 \times 180 \times 142$ for the model and $480 \times 512 \times 360 \times 142$ for the image. Assuming that each point can be stored using one bit, the memory space required to store them in 4D arrays is 785.2 Gbytes for the model pattern and $1,570.4$ Gbytes for the image pattern, while the memory space required to store their distance transforms is even larger.

Computation efficiency. Computing the distance transform in four dimensions using mask based algorithms, such as the one in [1], is significantly more expensive, in terms of time and memory, than the two-dimensional case. Given the size of the 4D space, this problem becomes particularly critical.

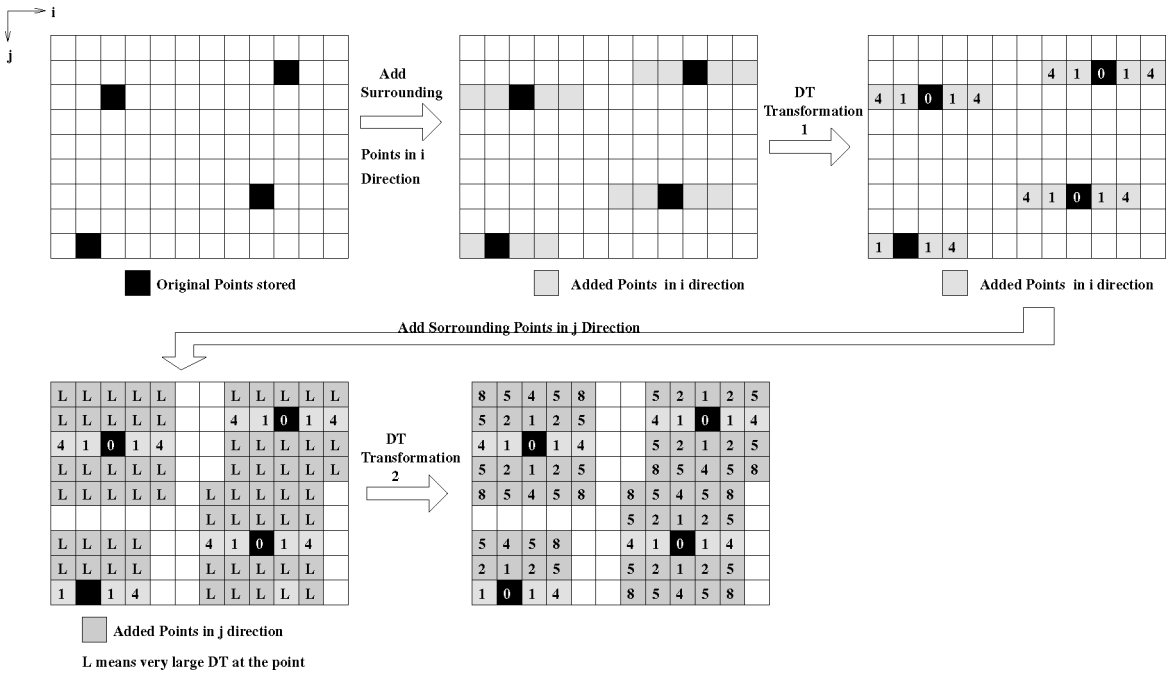


Fig. 5. Computation of the distance transform for sparse data.

The solution to these issues is to exploit the sparseness of the data and to use a distance transformation algorithm whose run time grows linearly with the domain dimensions, as discussed below.

5.2.1 Memory Management

As was illustrated above, it is not practical to use a 4D array to store the patterns due to the dimensions of the transformed domain. However, since we are dealing with segments instead of points, the actual data is very sparse. Thus, we propose storing *only* the data points and sufficiently large neighborhoods around these points. All other points in the space are considered too far from any data point to be of any interest and, therefore, are assigned a distance transform of infinity. This is illustrated in Fig. 3 where (a) shows a 2D slice of a distance transform with only four data points (the points with zero distance transform are marked in black), (b) shows the data points surrounded by their close (distance transform less or equal to 8) neighbors, and, finally, (c) shows that all the other points (distance transform greater than 8) are assumed to have a distance transform of infinity.

5.2.2 Computing the 4D Distance Transform

Obtaining the Hausdorff distance requires the computation of the distance tranform (DT) for the model and the image patterns. There are several approaches to compute the 2D distance transform [2], [5], [3], [10], [9], [12]. The commonly used chamfer methods [2], [5], [10] simply template a mask over the image. The parallel version of this method performs the calculations iteratively to update the result. The serial version of the algorithm requires two passes of the masks (left to right, then top to bottom).

On the other hand, multidimensional distance transforms (3D and above) are more complicated, due to memory and speed requirements. Borgefors presented algorithms for 3D and 4D distance transforms in [1]. The algorithms take different number of masks depending on the dimensions, and perform the serial computations in forward and reverse template fashion along each axis. Two masks consisting of five planes are needed for the four-dimensional n-neighbor and chamfer cases. The forward mask has to move in the increasing direction of each axis, while the backward mask does it in the reverse way. Therefore, there are a total of eight times of updated computations for the final DT of each rexel (4D point). Other algorithms are the one by Mullikin [11] extending the Euclidean DT of Danielsson's type [5] to the three-dimensional space with improved accuracy of distance values, and the one presented by Ragnemalm [13] which suggests a suboptimal scan algorithm for arbitrary-dimensional pictures.

All the above algorithms use different masks or weighted matrices in local operations to propagate distance values. Therefore, they are not error-free. Furthermore, as the dimension of the space grows higher, they require a higher number of masks and scannings.

Saito and Toriwaki [17] proposed a method that obtains the exact Euclidean distance transform and the Voronoi diagram based on the exact Euclidean metric for an n-dimensional picture. This algorithm requires only an n-dimensional array to store input output pictures and a single one-dimensional array for a work area. It does not use any mask, it is easy to implement regardless of the dimension, and it is fast. Thus, this algorithm has been selected for the proposed approach. The basic steps of the algorithm are outlined below for the three dimensional case, without any loss of generality.

Consider the input image $F = \{f_{ijk}\}$ with $(1 \leq i \leq L, 1 \leq j \leq M, 1 \leq k \leq N)$. Then, the 3D distance

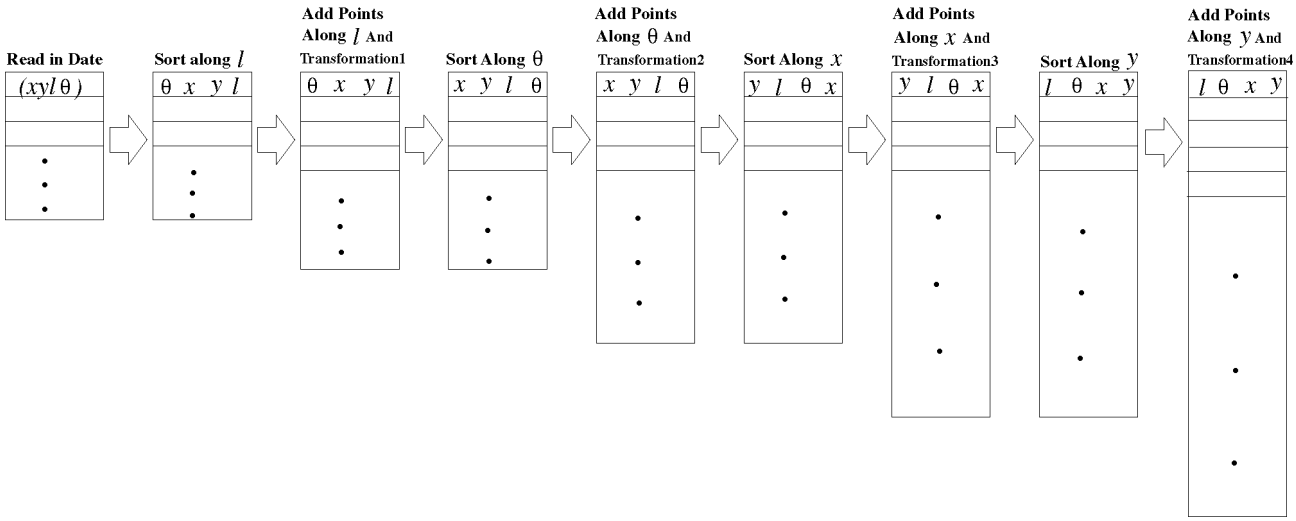


Fig. 6. Implementation of the distance transformation for sparse data. The list of points is sorted and expanded with neighbors in one direction before each transformation.

transform is obtained by applying the following three transformations:

Transform 1. Transformation in the i-axis direction. Derive from F a new image $G = \{g_{ijk}\}$:

$$g_{ijk} = \min_x \Big\{ (i-x)^2; f_{xjk} = 0, 1 \le x \le L \Big\}. \tag{5}$$

Transform 2. Transformation in the j-axis direction. Derive from the above image G a new image $H = \{h_{ijk}\}$:

$$h_{ijk} = \min_y \Big\{ g_{iyk} + (j-k)^2; 1 \le y \le M \Big\}. \tag{6}$$

Transform 3. Transformation in the k-axis direction. Obtain an image $S = \{s_{ijk}\}$ from the above image H:

$$s_{ijk} = \min_z \Big\{ h_{ijz} + (k-z)^2; 1 \le z \le N \Big\}. \tag{7}$$

In [17], it is shown that the obtained image $S = \{s_{ijk}\}$ is the squared Euclidean DT of the image $F = \{f_{ijk}\}$. In other words, a voxel (i, j, k) in the image S has a value equal to the square of the Euclidean distance from the voxel (i, j, k) to the closest data voxel in the image F.

Fig. 4 illustrates how the algorithm works. The top of the figure shows the result of the first transformation for the kth slice of a three-dimensional image with foreground pixels marked in black (with distance 0). The bottom of the figure shows the computations necessary to carry out the second transformation for pixel $(6, 6, k)$.

Computing the 4D Distance Transform for Sparse Data. Saito and Toriwaki's algorithm [17] can be easily adapted to compute the distance transform *only* at the data points and their neighbors for the case when the data is sparse. The only required modification is that, at each transformation, only the points considered at the previous transformation and their neighbors in the direction of the current transformation should be considered, starting with the data points in the first step. The new points, as well as all other points, are assumed to have an infinity distance transform. The procedure is illustrated in Fig. 5 for the image shown in the top left corner with four isolated data points marked in black. The first step adds neighbors in the horizontal direction; the second step computes the first distance transformation at these points; the third step adds the vertical neighbors of all the points considered in the second step; finally, the last step shows the result of applying the second distance transformation to these points. The number of neighbors added at each side must be large enough such that all the points with distance transform less than the selected threshold are considered. The modified algorithm can be efficiently implemented by sorting the set of data points and their neighbors before each transformation. For example, before the transformation for the θ axis is performed, the points must be sorted by $x_m, y_m, \log l$, and θ, such that all points with same values of x_m, y_m and $\log l$ are grouped together. These steps are summarized in Fig. 6.

5.2.3 Computing the Hausdorff Distance

The 4D transform obtained using the algorithm described in the previous section is used to compute the Hausdorff distance as a function of translation. Suppose I and M are the sets of 4D line feature points and their surrounding points in the image and the model, respectively. For a given translation $t = (x, y, l, \theta)$ and fractions f_1 and f_2 of the image and model line feature points to be considered, respectively, let K be the f_1 fraction of the total image feature points, and L be the f_2 fraction of the total model feature points. Then, the directed generalized partial distances for the model and the image are given by

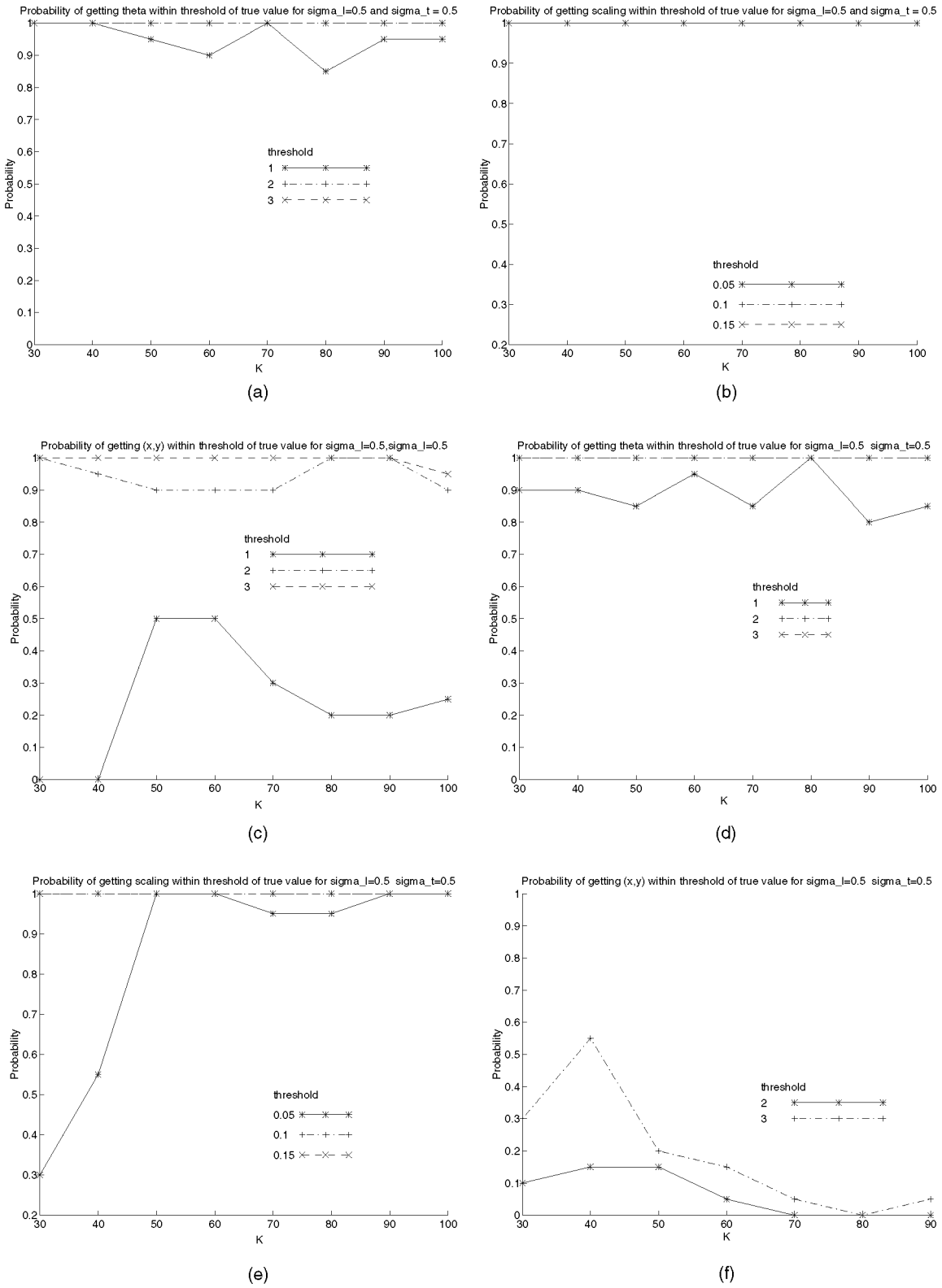


Fig. 7. Determination of factor K_s. Probability of correctly detecting using the 4D matching algorithm (a) rotation angle, (b) scaling factor, and (c) translation. Probability of correctly detecting using the 2D-2D matching algorithm (d) rotation angle, (e) scaling factor, and (f) translation.

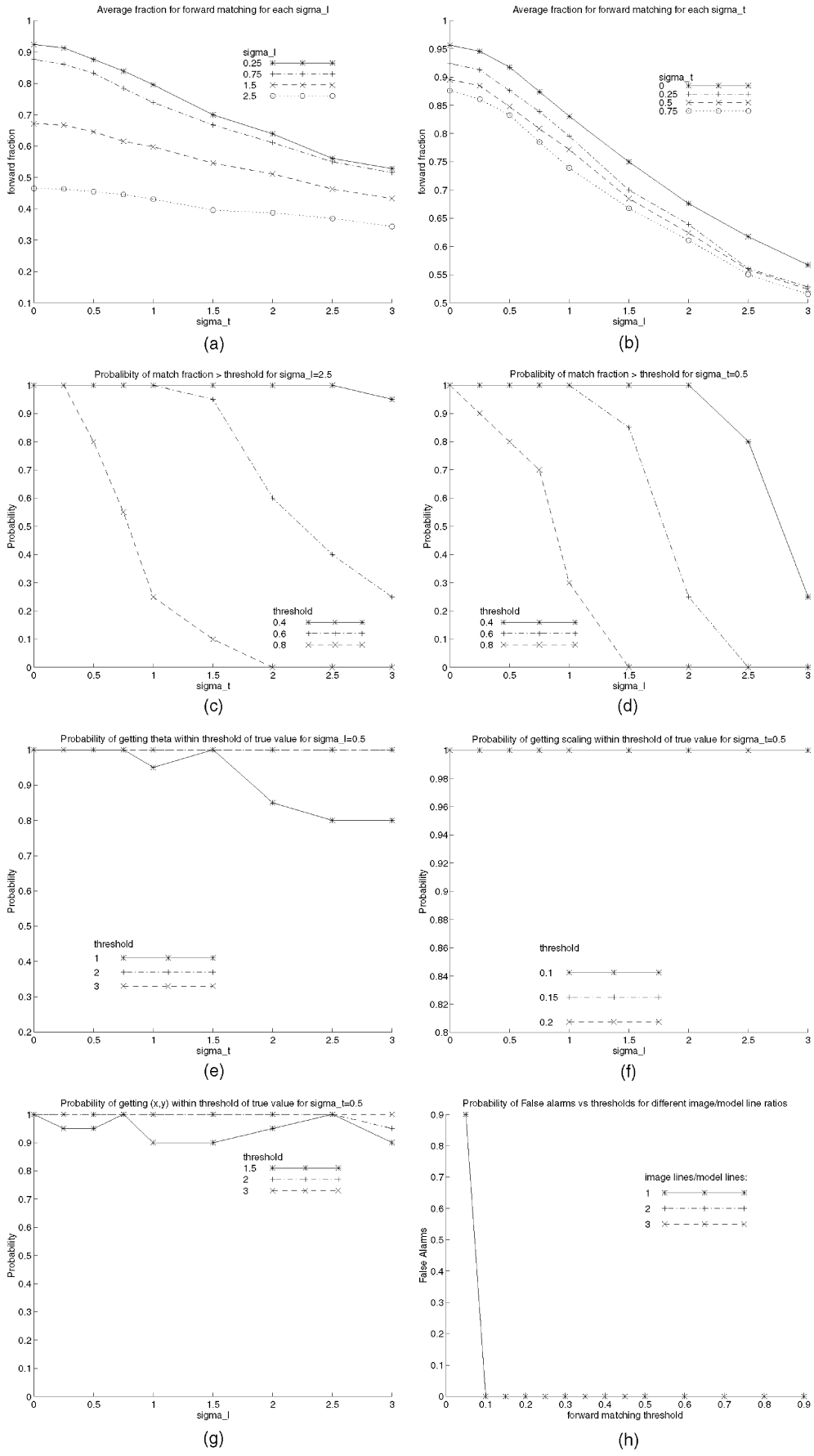


Fig. 8. The simulation results from the 4D matching (a to h).

$$\begin{aligned} D^M[x,y,l,\theta] &= h_K(t(M), I) \\ &= K^{th}_{m\in M}\{4DT(i) \\ &\quad \| \; i \in I \;, 1 = m + (x,y,l,\theta)\} \\ D^I[x,y,l,\theta] &= h_L(I, t(M)) \\ &= L^{th}_{i\in I}\{4DT(m) \\ &\quad \| \; i \in I \;, = \; i \; + (x,y,l,\theta)\}, \end{aligned}$$

where $4DT(.)$ represents the four-dimensional distance transform. Finally, the generalized partial Hausdorff distance is given by

$$\begin{aligned} D[x,y,l,\theta] &= H_{LK}(I, t(M)) \\ &= \max\big(D^I[x,y,l,\theta], D^M[x,y,l,\theta]\big). \end{aligned}$$

5.2.4 Searching Efficiently

The number of features required to describe the model and the image using line-based features is significantly smaller (by at least an order of magnitude) than the number of features required when using point-based features such as edgels. Thus, the time required to compare a transformation of the model with the given image is significantly less when the proposed 4D method is used than when point-based methods such as [8], [16] are used. Further efficiency can be attained by extending pruning techniques, such as "ruling out circles," "skipping forward" [8], and "cell subdivision" [16], derived for the point-feature-based algorithm to be used in four dimensions.

Ruling Out Circles. The slope of $D^M[x,y,l,\theta]$ cannot exceed 1—i.e., the function does not decrease faster than a linear relationship along any direction. That is, if $D^M[x_1,y_1,l_1,\theta_1] = d$, then $D^M[x,y,l,\theta]$ cannot be less than threshold τ in a circle of radius $d-\tau$ about (x_1,y_1,l_1,θ_1), where $d > \tau$. Thus, it is possible to jump out of the circle of radius $d-\tau$ without loosing a possible translation candidate. On the other hand, $D^I[x,y,l,\theta]$ does not necessarily hold the same property due to the fact that only a small window around the model is being considered. The proof of this property is a trivial generalization of the 2D case proven in [8].

Skipping Forward. This technique follows the ruling out circles claim. However, instead of jumping out of a circle, the algorithm skips along one axis (one row). In our algorithm, it is chosen as the $+y$ direction.

Following the notation introduced in [8], let D^M_{+y} be the distance in the increasing y direction to the nearest location where $D^M < \tau$ and ∞ if there is no such location:

$$D^M_{+y}[x,y,l,\theta] = \min_{\Delta y \geq 0, D^M[x,y+\Delta y,l,\theta] \leq \tau} \Delta y \; .$$

and let $D^M_K[x,y,l,\theta]$ be given by

$$D^M_K[x,y,l,\theta] = K^{th}_{m\in M} D^M_{+y}[m + (x,y,l,\theta)] \; .$$

Then, if $D^M_K[x,y,l,\theta]$ is 0, K of the values of D^M_{+y} probed must have been 0, and K of the probed values of D^M would be less or equal than τ. Also, if $D^M_K[x,y,l,\theta] = \Delta y > 0$, then $D^M[x,y,l,\theta] > \tau$ and $D^M[x,y+1,l,\theta], \ldots, D^M[x,y+\Delta y - 1,l,\theta] > \tau$ [8].

Cell Subdivision. The pruning technique of "cell subdivision" recently introduced in [16] can also be extended to the proposed 4D approach. A "cell" of translations $t = (t_x, t_y, t_l, t_\theta)$ is defined by two translation vectors, $t^l = (t^l_x, t^l_y, t^l_l, t^l_\theta)$ and $t^h = (w,w,w,w) + t^l$, with w a positive integer, such that $t^l \leq t \leq t^h$, component-wise. The *box distance transform* of the image I for a box of size w^4 is defined [16] as:

$$D'_{w^4}[x,y,l,\theta] = \min_{0\leq x',y',l',\theta' \leq w} D^I[x+x', y+y', l+l', \theta+\theta'],$$

where any portions of the distance transform $D^I[x,y,l,\theta]$ outside the boundary of the image array is treated as being infinite.

Let m be a point and $\mu = t^l[m], t[m]$ and $t^h[m]$ its translations using the vectors t^l, t, and t^h, respectively. The value $D'_{w^4}[\mu]$ is the minimum value that could be achieved by $D^I[t[m]]$, independent of the vector t, as long as it lies in the cell defined by t^l and t^h. Thus, the box distance transform can be used to rule out cells that do not have a large enough fraction of values of D'_{w^4} below the threshold τ. On the other hand, cells with a large enough fraction of low values of D'_{w^4}, can be subdivided into smaller ones to refine the translation. It must be noted that this technique, as the previous two, only eliminates transformations that are not optimal.

The price paid to use this technique is the table D'_{w^4}. While this table can be computed in $O(\log(w))$ passes through the distance transform D^I, it requires significantly more memory than the original D^I table, even for moderate values of w. This can be easily seen by considering an isolated data point in the image 4D domain and its neighbors with finite distance transform. Let d^4 be the number of points of this neighborhood (typically, $d = 3$). Then, the corresponding neighborhood in the box distance transform D'_{w^4}, has $(d + 2w - 2)^4$ points. In other words, the bigger the size of the cells, the less sparse the box distance transform is. Clearly, this limits the practical size of the cells that can be used with this technique.

6 Experiments and Performance Characterization

In this section, the method performance and sensitivity to segmentation problems are characterized using experimental protocol with simulated data designed to model clutter as well as broken and noisy lines. The algorithm is also tested with real images.

6.1 Experimental Protocol

The experimental protocol consists of: 1) ideal data generation; 2) noise models to generate perturbed data with annotated ground truth; and 3) tests for performance characterization. While this protocol was designed with the proposed matching algorithm in mind, it can be easily adapted to characterize any algorithm requiring image segments.

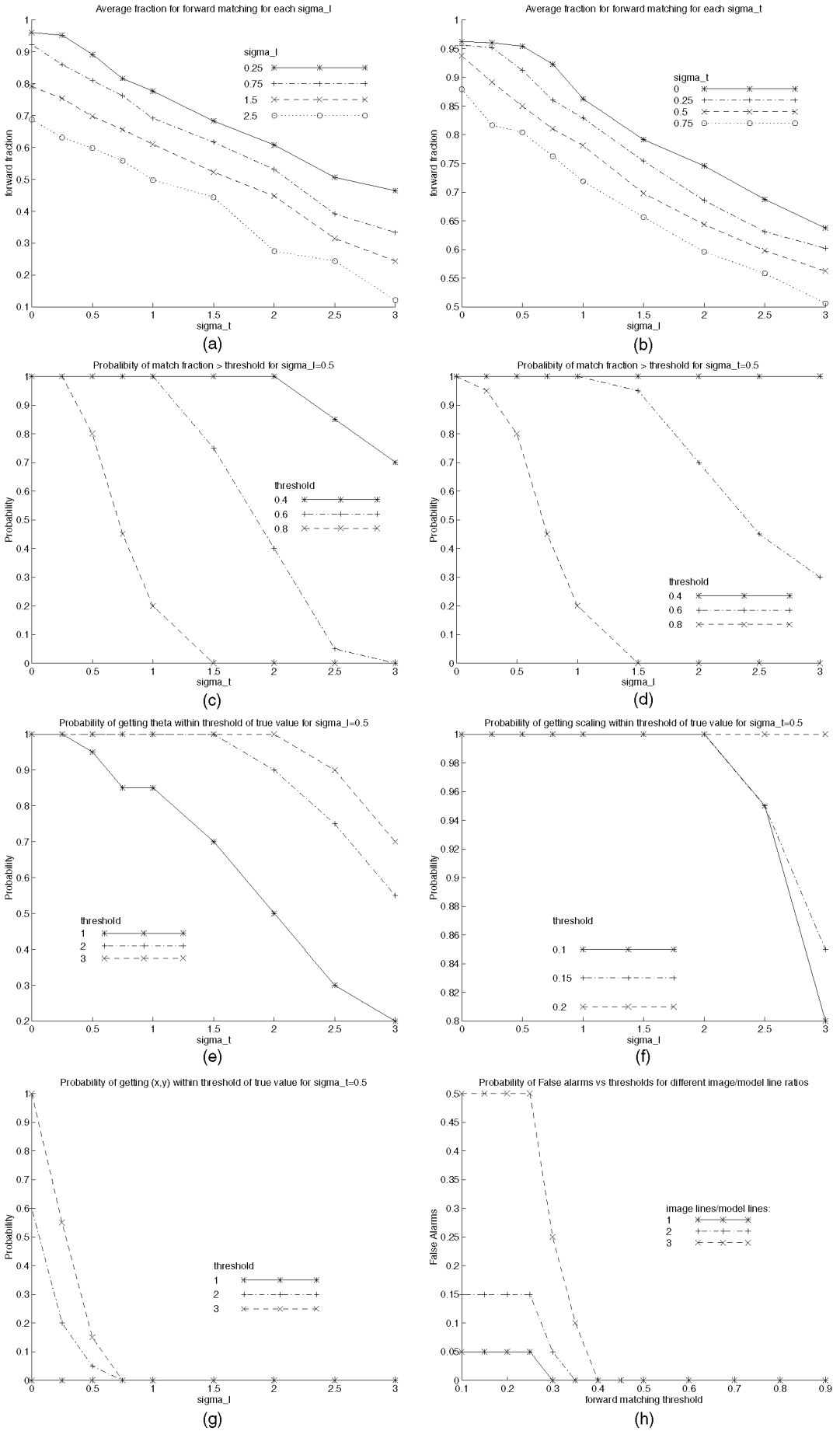


Fig. 9. The simulation results from the 2D matching algorithm (a to h).

6.1.1 Simulated Data

Model Generation: A set of lines (30) with random orientations, lengths, and locations are generated for the model such that the coordinates of their mid-points are uniformly distributed within a 512×512 image, their orientations are uniformly distributed between 0° and 180°, and their lengths are normally distributed with mean length of 70 pixels and a variance of 45 pixels.

Image Generation: Two types of images are generated: images containing a model and clutter and images containing clutter only. The images with a model are generated by rotating, scaling, and translating the model segments to a known location. There is no restriction in the selection of the rotation angle and the scaling factor. In our experiments, we used a rotation angle of 30° and a scaling factor of 0.8. Different types of noise perturbations are then added to the resulting set in order to generate the image line segments. The clutter-only images are obtained by generating random line segments uniformly distributed as described below.

6.1.2 Noise Perturbations

The following noise perturbations are used to simulate segmentation problems:

Clutter: A fixed percentage (100 percent) of random line segments are added to the image as outliers. Lengths of these lines are uniformly distributed between 0 and 128. Orientations are also distributed uniformly between 0° and 180°. Locations of the mid-points are uniformly distributed in the image except for the area where the model is expected to lie. A fixed smaller percentage (2 percent) of lines are uniformly added to that area later.

Length Uncertainty: A Rayleigh distribution with parameter σ_l is used to model the uncertainty in the length of the lines.

Broken Lines: The number of points where a line is broken is modeled using a Poisson distribution with parameter p_b. The points where the line is to be broken are then obtained using a uniform distribution along its length. The length of the gap to be created is obtained using a Rayleigh distribution.

Orientation Uncertainty: The orientation of the line is perturbed using a Gaussian distribution with zero mean and standard deviation of σ_t/l, where l is the length of the line. This is done since longer lines are expected to have less angular perturbation than shorter lines.

6.1.3 Determination of the Factor K_s

The factor K_s should be selected such that the probabilities of correctly detecting the rotation angle, the scaling factor, and the translation are maximized. These probabilities are obtained for $\sigma_t = \sigma_l = 0.5$ and varying K_s.

6.1.4 Performance Characterization

The performance of the 4D and 2D-2D matching algorithms is characterized in terms of the accuracy of the computed translation, rotation, and scaling transformations. Let $f(x_1, y_1, \theta_1, L_1), f(x_2, y_2, \theta_2, L_2) \ldots f(x_n, y_n, \theta_n, L_n)$ be the matching fractions from the forward or reverse Hausdorff matching results when noise is added to the image, where $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ are the corresponding translations in the x and y directions, $\theta_1, \theta_2, \ldots, \theta_n$ are the rotation angles, and $L_1, L_2, \ldots, L_n$ are the logarithms of the scale factors S times K_s. The transformation with the maximum fraction is the one selected:

$$(\hat{x}, \hat{y}, \hat{\theta}, \hat{L}) = \max_{i=1,n} f(x_i, y_i, \theta_i, L_i)$$

The following observations are made:

1. The selected forward matching fraction is averaged for 20 simulations and plotted against the rotation perturbation σ_θ and the length perturbation σ_l.
2. The probability of the selected forward matching fraction being greater than a threshold (0.4, 0.6, and 0.8) is found and it is plotted against the noise parameters σ_t and σ_l.
3. A plot is made for the probability of successful detection of the rotation angle against σ_t. A rotational angle is said to be successfully detected if the difference between the estimated angle $\hat{\theta}$ and the ground truth θ_{true} is less than a threshold (one, two, and three degrees).
4. A plot is made for the probability of successful detection of the true translation against σ_l. The detection is successful when $|f(\hat{x}, \hat{y}) - f(x_{true}, y_{true})| < \text{threshold}$, where x_{true} and y_{true} are the ground truth translation coordinates and the threshold was taken as 1.5, 2, and 3.
5. A plot is obtained for the probability of successful detection of the scaling against σ_l. A scaling factor is said to be successfully detected if the difference between the estimated scaling and the ground truth is less than a threshold (0.1, 0.15, and 0.2).
6. The probability of a false match is also studied by matching models against clutter, only images. The ratio between the number of lines in the model and the clutter image is taken as 1, 2, and 3 for this experiment. A false match is said to happen if the matching fraction is found to be greater than a given threshold. The results of the experiments are averaged for 20 simulations and plotted as a function of the threshold.

All of the above experiments are conducted for the 2D-2D and the 4D matching algorithms in order to compare their performance.

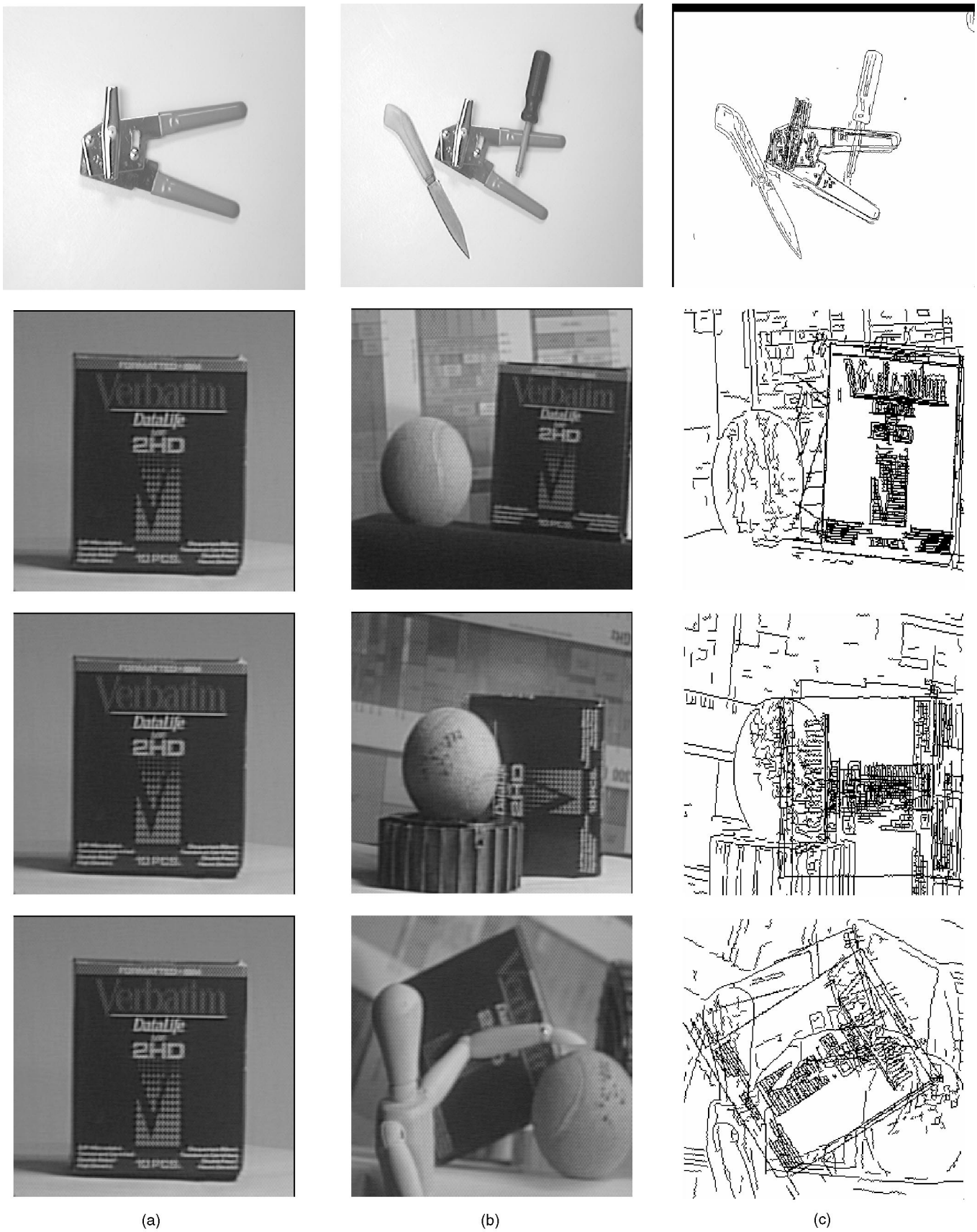
6.2 Results with Simulated Data

6.2.1 Determination of Factor K_s

The plots of the probabilities of correctly detecting the rotation, scaling, and translation transformations as K_s are varied are shown in Figs. 7a, 7b, and 7c for the 4D matching algorithm and in Figs. 7d, 7e, and 7f for the 2D-2D matching algorithm. It is seen that there is no single K_s value that can maximize all three probabilities. As a compromise, we chose $K_s = 60$ for the rest of the experiments.

6.2.2 Performance Characterization

All the plots resulting from conducting the experiments described in Section 6.1.4 are shown in Fig. 8 and Fig. 9 for the 4D and 2D-2D matching algorithms, respectively. It is



(a) (b) (c)

Fig. 10. Examples using real images. The running time for the first example was under 550 seconds on a SUN SPARC 5 and for the last three examples was under 400 seconds on a SUN Ultra 1.

observed from the results for the 4D matching algorithm that:

1. The matching fraction averaged for the 20 simulations plotted against the rotation perturbation and the length perturbation σ_l in Figs. 8a and 8b decreases as the noise deviations σ_l and σ_t increase.
2. The probability of the matching fraction being greater than thresholds of 0.4, 0.6, and 0.8 plotted against the noise parameters σ_t and σ_l are shown in Figs. 8c and 8d. It is seen that the probability decreases as the noise deviations σ_t and σ_l increase. However, the probability decreases faster with respect to σ_t. This is expected because if a line is deviated from its ideal orientation, most of the line points deviate away from the ideal positions. However, if there is some uncertainty for the line length, a large number of points may still stay at their ideal positions.
3. Figs. 8e and 8f show the relation between the probability of the angle θ being within a threshold of the true rotation angle against σ_t and the relation between the probability of getting the scaling within the threshold of the true scaling value and σ_l, respectively. It is seen that the probability of obtaining the true scaling is very robust to the noise (σ_t).
4. Fig. 8g shows the probability of the translation being within threshold of the true translation against σ_l. It is seen that the probability of computed translation within the true translation is also very robust with respect to σ_l.
5. Finally, Fig. 8h shows the plots for the false matches when the ratio between the number of segments in the image and the model were one, two, and three (for images without a model).

Comparing the above results to the ones obtained when using the $2D-2D$ matching algorithm shown in Figs. 9a, 9b, 9c, 9d, 9e, 9f, 9g, and 9h, it can be observed that:

1. The matching fraction in the $4D$ case is more robust to σ_t than in the $2D$ case; however, there is no significant improvement with respect to σ_l.
2. All the probabilities of obtaining the correct translation, rotation and scaling in the $4D$ case are much more robust than in the $2D$ case.
3. Finally, the false alarms in the $4D$ case decrease much faster to the increase of the matching threshold than in the $2D$ case.

6.3 Results with Real Data

The algorithm was also tested using real images taken in our laboratory. Rotation and translation were modeled by moving the camera with a robot arm, and scaling was modeled by zooming the camera. Fig. 10 shows four examples, where the left column corresponds to the models, the center column to the given images, and the last column to the alignment results, respectively.

The first example is an image with relative little clutter and some occlusion. The second example does not present occlusion, but it has heavy background clutter with many distracting linear features. The third and fourth examples have similar background to the previous one, but the model is moderately occluded and severely occluded, respectively. The obtained model alignments are in close agreement with the given images, except for the last one. The misalignment in the fourth case happened because many of the lines corresponding to the model in the given image were broken due to severe occlusion. The first image is 512×480 and run in less than 550 seconds on a SUN SPARC 5 station, while the remaining three are 256×256 and run in less than 400 seconds on a SUN Ultra 1.[1] The above execution times where attained without using "cell subdivision" and it is expected that if it were used, the running time would be less. Finally, it is important to notice that, since we are actually dealing with sparse line features and we are using a sparse data structure in our implementation, the running time is rather a function of the number of lines than of the image size.

7 Conclusions

In this paper, a method for model-based recognition using line features and a four-dimensional Hausdorff distance was presented. The proposed method separates rotation, scaling, and translation into three parts and associates them with the line's orientation, length, and middle points, respectively. The original image is transformed into a 4D new domain where the rotation and scaling transformations in the original domain are mapped into a translation in the new domain. As a result, the rotation, scaling, and translational relationship between the image and model can be obtained simultaneously using the four-dimensional Hausdorff distance. Computational efficiency is achieved by exploiting the sparseness of the proposed representation. Furthermore, this method greatly improves the false alarm rate obtained in our previously developed 2D-2D matching algorithm.

The performance of the method and its sensitivity to segmentation problems were characterized by using a rigorous experimental protocol modeling the most common types of errors in segment extraction. The algorithm performs well, degrading smoothly as the perturbations increase. The comparisons between the 4D matching algorithm and previous 2D matching algorithm show that 4D matching algorithm is more robust to the noise and it has smaller false alarm rate. The method was also tested on real images with and without partial occlusion and in the presence of clutter with runing time significantly less than previous methods.

Appendix

Algorithms

BEGIN: Computation of Distance Transform

Read in all the image and model line segments, generate line feature points;

1. As a comparison, times reported in [15] are between 2,379 and 19,780 seconds, and times reported in [16] are 3,166 and 14,726 seconds for images of similar complexity as the last three examples shown here.

Part 1: Add the surrounding points and calculate DT
  For each axis, **do**:
    Sort the points;
    Add surrounding points in the corresponding direction;
    Calculate partial distance transformations;
    Obtain the final DT for four-dimensional feature points:
  End do
Part 2: Calculate D_{+y}
  For each of the 4D points, **do**:
    Find the set R of points with the same x, l, θ
    Within R, **do**:
      Forward D_{+y} calculation;
      Backward D_{+y} calculation;
    End do
  End do
Part 3: Hash table for image points/surrounding points
  Allocate the memory;
  For each point, **do**
    Hash address = hash_function(x, l, θ);
    Add this point to the link list of this address;
  End do
END: Computation of Distance Transform.

BEGIN: Model and Image Matching
Part 1: Find translations for the forward matching
  For scaling from S_{min} to S_{max}, **do**:
    For θ from 0 to 180, **do**:
      Rotate_Scale_Model;
      For translation x from 0 to x_{max}, **do**:
        Set translation $y = 0$;
        Set minimun $D_{y_{min}}$ to a large value, $D_{y_{min}} = y_{max}$;
        While translation $y < y_{max}$, **do**:
          For each point in the image/surrounding list, **do**:
            Match_model_pts to the image/surrouding pts, get D_y;
            If $D_y > 0$, **Then** $D_{y_{min}} = \min(D_y, D_{y_{min}})$;
            If no match or $D_y > th_1$, **Then** increment no match COUNT;
            If $COUNT > th_2$, **Then** break and go to next y;
            Else go to the next element;
          End do
          If $COUNT < th_2$, **Then** write to the translation list;
          Increment $y = y + D_{y_{min}}$;
        End while
      End do
    End do
Part 2: the backward matching
  For each translation (x, y, l, t) obtained above
    compute $D_I[x, y, l, t]$
    obtain $D[x, y, l, t]$ defined in the text
  End do
END: Model and Image Matching

Details for Match_Model_pts function:
  Let y be the current translation;
  Input model point: Key;
  Hash Key from the image/surrounding point lists;
  Search through the list until x, l, and θ are the same;
  If cannot find a match, **Then**:
    $D_y = y_{Max} - (Key \rightarrow y + y)$;
    Return D_y and no match signal;
  Else
    Starting from the list, find how many pts have the same x, l, and θ, say it is K_r
    Let $y_m = Key \rightarrow y + y$;
    For $i = 0$ to K_r, **do**:
      If $y_m < y(\text{list element})$, **Then**:
      $D_y = y(\text{list element}) - y_m$;
      Return no match signal;
      End if
      If $y_m == y(\text{list element})$, **Then**:
      $D_y = D_{+y}(\text{list element})$;
      Return match signal;
      Else, increament i;
    End do

ACKNOWLEDGMENTS

The authors would like to thank Dr. Huttenlocher for providing the software for point-based Hausdorff distance matching, Tarak Gandhi for his help in implementing some of the source code and designing the experimental protocol, and Dr. Sznaier for many fruitful discussions. The authors would also like to thank the anonymous reviewers for suggesting the use of cell subdivision to speed up the algorithm and for making comments on how to make the paper more clear.

REFERENCES

[1] G. Borgefors, "Distance Transformations in Arbitrary Dimensions," *Computer Vision Graphics and Image Processing: Image Understanding*, vol. 27, pp. 321–345, 1984.
[2] G. Borgefors, "A New Distance Transformation Approximating the Euclidean Distance," *Proc. Conf. Eighth Information Center for Physics Research*, pp. 336–339, 1986.
[3] G. Borgefors, "Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm" *Int'l J. Computer Vision*, vol. 10, no. 6, pp. 849–865, 1988.
[4] A. Califano, R. Mohan, "Multidimensional Indexing for Recognizing Visual Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 373–392, Apr. 1994.

[5] P.E. Danielsson, "Euclidean Distance Mapping," *Computer Graphics Image Processing,* vol. 14, pp. 227–248, 1980.
[6] M. Dubuisson and A.K. Jain, "A Modified Hausdorff Distance for Object Matching," *Proc. Int'l Conf. Pattern Recognition,* pp. 566–568, 1994.
[7] D.P. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing Images Using the Hausdorff Distance under Translation," Technical Report 1211, Dept. of Computer Science, Cornell Univ., 1991.
[8] D.P. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing Images Using the Hausdorff Distance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 9, pp. 850–863, Sept. 1993.
[9] A.V. Karzanov, "Quick Algorithm for Determining the Distance from the Points of the Given Subset of an Integer Lattice to the Pionts of its Complement," *Cybernetics and System Analysis,* pp. 177–181, Apr.–May, 1992.
[10] F. Leymarie and M.D. Levine, "Fast Raster Scan Distance Propogation on the Discrete Rectangular Lattice," *Computer Vision Graphics and Image Processing: CVGIP: Image Understanding,* vol. 55, pp. 84–94, 1992.
[11] J.C. Mullikin, "The Vector Distance Transform in Two and Three Dimensions," *Computer Vision Graphics and Image Processing,* vol. 27, pp. 526–535, 1992.
[12] D.W. Paglieroni, "Distance Transform: Properties and Machine Vision Applications," *Computer Vision Graphics and Image Processing,* vol. 54, pp. 56–74, 1990.
[13] I. Ragnemalm, "The Euclidean Distance Transform in Arbitrary Dimensions," *Pattern Recognition Letters,* vol. 14, pp. 883–888, 1993.
[14] W. Rucklidge, "Efficient Computation of the Minimum Hausdorff Distance for Visual Recognition, Technical Report, Dept. of Computer Science, Cornell Univ., 1995.
[15] W. Rucklidge, "Locating Objects Using the Hausdorff Distance," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 457–464, 1995.
[16] W. Rucklidge, "Efficiently Locating Objects Using the Hausdorff Distance," *Int'l J. Computer Vision,* vol. 24, no. 3, pp. 251–270, 1997.
[17] T. Saito and J. Toriwaki, "New Algorithms for Euclidean Distance Transformation of an n-Dimensional Digitized Picture with Application," *Pattern Recognition,* vol. 27, no. 11, pp. 1,551–1,565, 1994.
[18] X. Yi and O.I. Camps, "Line Feature-Based Recognition Using Hausdorff Distance," *Proc. Int'l Symp. Computer Vision,* pp. 79–84, Nov. 1995.
[19] J. You, E. Pissaloux, J.L. Hellec, and P. Bonnin, "A Guided Image Matching Approach Using Hausdorff Distance with Interesting Points Detection," *Proc. Conf. First Image Processing,* vol. 1, pp. 968–972, 1994.

Xilin (Steven) Yi received the BSEE degree from Beijing University in 1988, the MSEE degree from Tsinghua University, China, in 1991, the MS degree in physics from Temple University, Philadeplphia, Pennsylvania, in 1993, and the PhD in electrical engineering from The Pennsylvania State University, Pennsylvania, in 1997. He joined the Intelligent Machine Technology Division of ENSCO, Inc. as a machine vision staff scientist in 1997. At ENSCO, Inc. Dr. Yi has been responsible for many machine vision project designs such as food quality control, lumber grading, postal mail tray sorting, and railway air hose inspection. He is also a board member of Web2Click, Inc., an online company that provides Internet web design/service, web hosting, and community chat center. Dr. Yi has a broad interest in science and engineering, especially in machine vision, imaging processing, and internet renovation. Dr. Yi is a member of SPIE and a member of the IEEE.

Octavia I. Camps received the BS degree in computer science and the BS degree in electrical engineering from the Universidad de la Republica (Uruguay) in 1981 and 1984, respectively, and the MS and PhD degrees in electrical engineering from the University of Washington, Seattle, in 1987 and 1992, respectively. In 1992, she joined the faculty at The Pennsylvania State University, where she is currently an associate professor in the Department of Electrical Engineering and the Department of Computer Science and Engineering, and a co-director of the Center for Intelligent Information Processing (CIIP). Dr. Camps was awarded the SWE Outstanding Female Engineering Student Award in 1988, a GTE Fellowship Award in 1990, and A United States NSF Research Initiation Award in 1993 for her work on robust 3D object recognition. Her current research interests include object recognition, active vision, reverse engineering systems, image processing, and pattern recognition. Dr. Camps is a member of the IEEE Computer, Robotics and Automation, and Signal Processing Societies, the ASEE, and Tau Beta Pi.