Tracking Non-Rigid Objects in Complex Scenes *

Daniel P. Huttenlocher   Jae J. Noh   William J. Rucklidge

Computer Science Department
Cornell University
Ithaca, NY 14850

Abstract

We describe a model-based method for tracking non-rigid objects moving in a complex scene. The method operates by extracting two-dimensional models of an object from a sequence of images. The basic idea underlying the technique is to decompose the image of a solid object moving in space into two components: a two-dimensional motion and a two-dimensional shape change. The motion component is factored out and the shape change is represented explicitly by a sequence of two-dimensional models, one corresponding to each image frame. The major assumption underlying the method is that the two-dimensional shape of an object will change slowly from one frame to the next. There is no assumption, however, that the two-dimensional image motion between successive frames will be small.

1 Introduction

The problem of tracking moving objects has received a good deal of attention in the computer vision community over the last few years (e.g., [2, 3, 5, 6, 7, 8, 9]). In this paper we focus on the problem of tracking non-rigid objects in complex scenes, including the case where there are other moving objects present. We use a model-based method in which two-dimensional models are extracted from the image data, and matched to successive frames of the image sequence. Because a model-based approach is able to exploit global attributes of the object being tracked, it can provide significant advantages over purely local methods for situations in which the environment is cluttered, there are multiple moving objects, or there may be a large motion of an object from one frame to the next. A number of other researchers have taken a model-based approach to motion tracking (e.g., [5, 6, 7]). What characterizes our approach is that there is no constraint on where in the image an object may have moved from one frame to the next, and there may be multiple moving objects in the scene. Moreover, the model of an object is acquired dynamically from the image sequence, rather than being provided a priori.

The central observation underlying our method is the fact that the two-dimensional image of an object moving in three-space can be decomposed into two parts

- a two-dimensional shape change, corresponding to a new aspect of the object becoming visible or an actual shape change of the object, and
- a two-dimensional motion (of some restricted kind) in the image, corresponding to the motion of the visible aspect of the object.

The main condition imposed by our method is that the two-dimensional shape of an object not change greatly between two successive frames of an image sequence. In particular, there is no assumption that the two-dimensional motion from one frame to the next be small. This is a less restrictive assumption regarding the nature of the motion than is made by local differential methods. Such methods must assume that the entire change in the object from one image to the next is small (local), whereas we only assume that the *shape* change is small, and not the motion.

We represent an object as a sequence of binary images, one corresponding to each frame of the input image sequence (which also consists of a sequence of binary images). Each frame of the model specifies a set of pixels in a given sub-region of the corresponding image frame (i.e., the model at time t is a subset of the image features at time t). These pixels constitute the appearance, or shape, of the object being tracked at that frame. Thus a model evolves from one

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

time step to the next, capturing the changes in the image shape of an object as it moves. In the current implementation, the input binary images are intensity edges extracted from a sequence of grey-level images. However, any other means of deriving binary features from an image sequence could be used.

The basic tracking method operates by comparing the model at a given frame, M_t, to the image at the next frame, I_{t+1}, in order to find the transformation specifying the best location of that model in that image (where the model and image are binary images derived by some feature extraction mechanism). Then a new model, M_{t+1}, is formed by selecting the subset of I_{t+1} that is 'near' the transformed model M_t. This new model, M_{t+1}, represents the shape of the object at the next time frame. The method of tracking an object is thus based entirely on comparing two-dimensional geometric structures, as represented by binary image models, between successive frames. There is no computation of local differential image quantities such as the optical flow or motion field. There are no a priori models of the object being tracked.

As an illustration of the method, Figure 1 shows the images and the corresponding models for six frames from a 100 frame sequence. Each entry in the figure contains an image frame (360 × 240 pixels), the intensity edges found using a method similar to [1], and the model extracted for that frame. The model at each time is a subset of the intensity edges for that frame. Each model is shown with a bounding box; note that the size of the models changes dynamically (as is described below). The third frame in the figure (at the bottom left) illustrates a situation where the object was changing shape very rapidly from one frame to the next, so while the object is still being tracked successfully, the representation of its two-dimensional shape is relatively degraded. Despite the fact that the object changes two-dimensional shape substantially in successive frames, and even more so during the entire sequence, it is tracked successfully through all 100 frames. (Note that the models shown in the figures are the result of tracking the entire sequence of 100 frames.)

The current implementation of the method requires about 5 seconds per frame for a half-resolution NTSC video image (360 × 240 pixels) on a SPARCstation-2 (not including edge detection, which is about another second per frame).

The major aspects of our approach are:

1. Decomposing the image of a moving 3D object into two parts: a 2D motion and a 2D shape change. The shape change is assumed to be relatively small from one frame to the next, but the motion can be arbitrarily large (i.e., there is no 'local search window' in the image).
2. Capturing the 2D shape change between successive images with 2D geometric models that evolve across time. The models provide global geometric constraints for tracking an object.
3. Fast 2D model matching using the minimum Hausdorff distance (a min-max-min distance described below). This distance measures proximity without computing an explicit correspondence.

In the next section we make precise the notion of a 'small' change in two-dimensional shape, and define the measure that we use to decompose the change in the 2D image of an object into a 2D motion and a 2D shape change. In Section 3 we describe how to use this notion of shape change to do tracking. In Section 4 we describe an implementation of the tracking method, and in Section 5 we discuss extensions of the method to handle multiple moving objects and objects that can disappear from view.

2 Comparing 2D shapes

We define a shape to be a two-dimensional geometric object which may be transformed by some given group of transformations. For example, a square oriented horizontally and a square oriented diagonally could be considered to be the same shape or they could not, depending on the allowable transformations. If the shapes are allowed only to translate, then they are different: one is a square and the other is a diamond. If they are allowed to undergo Euclidean motion (translation and rotation) then the two shapes are the same. More formally, given two geometric objects A and B, we say that they have the same shape exactly when there exists some $g \in G$ such that $g(A) = B$, where G is the allowable group of transformations.

In order to measure the difference between shapes, we use the minimum Hausdorff distance, under the action of the given allowable transformations, G. In the form that we use it here, we limit ourselves to finite point sets. For two point sets P and Q the Hausdorff distance between the sets is defined as

$$H(P,Q) = \max(h(P,Q), h(Q,P)) \qquad (1)$$

where

$$h(P,Q) = \max_{p \in P} \min_{q \in Q} \|p - q\|, \qquad (2)$$

Figure 1: Six selected frames (numbers 1,20,40,60,80 and 100) from a 100 frame motion sequence, and the corresponding models of the object being tracked. Each model is a subset of the edge pixels from that image frame.

and $\|\cdot\|$ is some norm for measuring the distance between two points p and q (which we will take to be the L_2 norm, or Euclidean distance). Thus this distance measures the degree to which each point of P is near some point of Q and vice versa. The distance is small when every point of one set is near some point of the other and vice versa.

The Hausdorff distance measures the difference between fixed point sets, whereas we are interested in measuring the difference between *shapes* of point sets. Given our definition of a shape as a point set modulo the action of some transformation group G, the natural definition of a distance is simply the minimum with respect to the action of that group,

$$D_G(P,Q) = \min_{g\in G} H(g(P),Q)\,. \qquad (3)$$

In other words, the difference between two shapes is the minimum distance between them under all possible transformations of one shape with respect to the other. Intuitively, this distance depends on the most mismatched point of one object with respect to the other, and vice versa. That is, there must be a transformation in G that brings all of one object near some part of the other object, and vice versa, in order for the distance to be small.

In practice, the fact that the Hausdorff distance measures the most mismatched point makes it very sensitive to the presence of any outlying points. Thus we use a more general rank order 'distance', which replaces the maximization operation in equation (2) with a rank operation (i.e., selection of the median value or some other quantile). A quantile is a more robust measure than the maximum, as is commonly known in statistics. This 'partial distance' is defined as

$$h_K(P,Q) = \mathop{K^{\mathrm{th}}}_{p\in P} \min_{q\in Q} \|p-q\|, \qquad (4)$$

where $K^{\mathrm{th}}{}_{p\in P}\, f(p)$ denotes the K-th ranked value of $f(p)$ over the set P. That is, if we consider the points in P to be in sequence ordered by their values $f(p_1) \le \ldots \le f(p_n)$, the K-th element in this sequence, $f(p_K)$, is the K-th ranked value. For example, the n-*th* ranked value is the maximum (the largest element in the sequence), and the $n/2$-*th* ranked value is the median.

The partial distance, $h_K(P,Q)$, identifies the subset of P of size K which has the smallest directed Hausdorff distance to the set Q. Intuitively, $h_K(P,Q) = d$ when there is some subset of P of size at least K such that each point in this subset is within distance d of some point in Q. Thus this measure allows for a portion of P to not correspond to anything in Q (as occurs, for example, with occlusion). For instance, if we compute $h_K(P,Q)$ with $K = \lfloor .75n \rfloor$ (i.e., using the 75-th percentile distance), then up to 25% of the points of P need not be near any points of Q (see [4] for more details).

One of the interesting properties of the Hausdorff distance, and the partial 'distance', is the asymmetry inherent in the computation. The fact that every point of P (or some given-sized subset of P) is near some point of Q says nothing about whether every point of Q (or some given-sized subset) is near some point of P. In other words, $h_{K_1}(P,Q)$ and $h_{K_2}(Q,P)$ can attain very different values. The maximum of these two values defines the partial Hausdorff distance.

3 Tracking using the Hausdorff distance

With the above-mentioned asymmetry of the Hausdorff distance in mind, let us now return to the tracking problem. The idea is to exploit the asymmetry in the distance to perform two different functions: (i) finding where the model at a given time, M_t, moved to in the image at the next time, I_{t+1}, and (ii) computing the new model, M_{t+1}, from M_t and I_{t+1}. For the remainder of the paper we will assume that the model M_t has m points (i.e., in the binary image representation of M_t there are m nonzero pixels).

3.1 Locating the object in a new image

The directed rank order 'distance' from the model M_t to the image I_{t+1}, which we will refer to as the *forward* distance, measures the degree to which some portion of the model resembles the image. The minimum value of this distance identifies the best position of M_t in I_{t+1}, under the action of some group G. This minimum value of the forward distance is given by

$$d = \min_{g\in G} h_K(g(M_t), I_{t+1}) = \min_{g\in G} \underset{p\in M_t}{K^{\text{th}}} \min_{q\in I_{t+1}} \|g(p)-q\|, \tag{5}$$

which identifies the transformation $g^* \in G$ of M_t that minimizes (4). That is, at least K of the m points of $g^*(M_t)$ are all within distance d of some point of I_{t+1}, and this is the minimum such distance (there is no $g \in G$ for which at least K points of $g(M_t)$ are less than d from some image point).

Intuitively, the forward distance identifies the best 'position', g^*, of M_t in the image I_{t+1}. Figure 2 illustrates the forward distance computation, where the transformation group G is the group of translations. The figure shows a model, an image, and the translation of the model that minimizes the partial directed Hausdorff distance for $K = \lfloor .8m \rfloor$. The distance at this minimizing translation is $\sqrt{2}$, meaning that at least 80% of the model pixels lie within one pixel (diagonally) of some image pixel.

3.2 Updating the object model

Having used equation (5) to identify the best location g^* of the model M_t in the subsequent image frame I_{t+1}, it now remains to build M_{t+1} by determining which pixels of I_{t+1} are part of the new model. We are interested in finding some subset of the image, I_{t+1} that 'agrees well' with the transformed model $g^*(M_t)$. We do this by using the distance from each point of I_{t+1} to the nearest point of $g^*(M_t)$ as a criterion for selecting the subset of image points that belong to M_{t+1}. That is we define,

$$M_{t+1} = \{q \in I_{t+1} | \min_{p\in M_t} \|g^*(p) - q\| \le \delta\}$$

for some distance δ. In other words, M_{t+1} is all those points of the image that are within distance δ of some point of $g^*(M_t)$.

The choice of the distance δ controls the degree to which the method is able to track objects that change shape. For instance, if $\delta = 0$ then only those pixels of I_{t+1} that are directly superimposed on $g^*(M_t)$ will be included in M_{t+1}, and thus the method will track only motions in G and not shape changes. In practice, setting $\delta = 0$ will cause the tracker to lose an object after several frames even if the object shape does not actually change, due to noise and uncertainty in the locations of pixels. The larger the value of δ, the more that the tracker is able to follow non-rigid motion, and to 'pick up' new parts of an object that have come into view since the previous frame. The parameter δ should be thought of as reflecting how much of the change from one frame to the next will be incorporated into the model (versus being left in the background). Thus, as δ becomes larger the ability to track non-rigid motion increases, at the cost of also possibly 'tracking' the background.

Often, in sequences showing people walking, the body and head change shape very little from one frame to the next, while the arms and legs move rapidly compared with the overall motion. The body and head are included in the models M_t since they tend not to change shape, while even if an arm, say, is in M_t, it may well not be included in M_{t+1} since it might have moved more than δ pixels. However, if an arm is not initially in M_t but is held stationary (or moves only slowly) relative to the body, it will gradually be reacquired over the next few frames (at a rate of about δ pixels per frame). This phenomenon can be seen in the models shown in Figure 1.

The models in Figure 1 were generated by running the tracker with K set to be 80% of the model pixels for the forward match (i.e., $K = \lfloor .8m \rfloor$), and with a δ of 8 pixels for constructing the models. The group of transformations, G, was the translation group. These parameter values were used successfully in both examples shown in the paper, and in many other sequences

Figure 2: Using the forward distance to locate a model in an image.

which we have analyzed with this tracker. In the following section we describe the tracker in more detail. We then consider some other techniques that are used to enhance the basic method. At a high level, however, the method is simple:

- Use the minimum directed partial Hausdorff distance from the model to the image to find where an object moved to. This tracks the two-dimensional motion of the object.
- Use the distance from the image to the transformed model as a criterion to select a subset of the image pixels that form part of the next model. This tracks the two-dimensional shape change of the object.

4 An implementation of the tracker

Recall that the basic observation underlying the method is to decompose the image of an object moving in space into a two-dimensional motion and a two-dimensional shape change. In the current system, we use the group of translations as the group of allowable two-dimensional motions. That is, any change in the image of an object other than a translation is encoded as a change in the two-dimensional shape of that object (i.e., a change in the model from one frame to the next). Allowing only translational motion means that rotation or scaling in the image are treated as changes in shape. While it is possible to allow larger groups of image motions (e.g., rotation and scale in addition to translation) there is a significant computational cost to doing so.

4.1 Finding the model's new location

Possible locations of the model M_t are identified in the image at the next time frame, I_{t+1}, by finding the translation giving the minimum forward distance as in (5). However, rather than computing the single translation giving the minimum distance, we identify the set of translations of M_t, call it X, such that the partial directed Hausdorff distance is no larger than some value τ:

$$X = \{x | h_K(M_t \oplus x, I_{t+1}) \leq \tau\}, \qquad (6)$$

where $\oplus$ is the Minkowski sum notation, $A \oplus x = \{a + x | a \in A\}$ for any set A and vector x. Intuitively, X is the set of possible locations of M_t in I_{t+1} — the translations for which at least K of the m points (nonzero pixels) of $M_t \oplus x$ are within distance τ of some nonzero pixel of I_{t+1}.

There are two reasons for finding all the translations where the distance is small, rather than the single best translation. First, there may be multiple translations that are essentially the same quality, in which case simply finding the best translation would not allow us to detect the presence of multiple matches. Second, there are efficient methods for finding all the translations such that the Hausdorff distance is less than some threshold (see [4]).

Once the set of possible translations, X, has been computed, it is partitioned into equivalence classes based on connected components in the grid of translations. This is because one instance of the object in the image may result in a number of translations that are below the threshold τ, and these are likely to all be neighboring translations. Thus we break X into sets X_i of neighboring translations (i.e., connected components in the translation-space where the distance is below τ). Each of these sets of neighboring translations corresponds to a possible instance of the object, and is processed separately. We find the best translation in each X_i, and use that translation as the representative possible location of the model for the set X_i.

For each translation $x \in X_i$ we know the actual distance from the model to the image at this translation, $d = h_K(M_t \oplus x, I_{t+1})$. This distance can be used to rate the quality of the translations in X_i, those with lower distance being better. Moreover, for each translation $x \in X_i$ we know the actual number of model points, s, $K \leq s \leq m$, that are within distance d of image points. In other words, the distance d tells us that at translation x at least K of the m model points are within distance d of image points, but there may be more than K such points (the actual number of such points being s). Therefore, each position $x \in X_i$ of the model is 'scored' by a pair $(d, s/m)$, where d is the forward distance $h_K(M_t \oplus x, I_{t+1})$ and s/m is the fraction of the model points that are within distance d of some image point.

We thus define the best match in each set X_i as the translation that minimizes d, which is only natural as we are seeking the minimum Hausdorff distance as given in equation (5). If there are multiple translations with the same (minimal) value of d then we select the one with the largest fraction s/m. This match is the 'representative' position of the model in the image for the equivalence class X_i. For the remainder of this section, we assume that there is just one equivalence class of translations, $X_1 = X$. In general this is the case, because multiple matches only occur when there are several objects of nearly the same two-dimensional shape in the image. In Section 5 we handle the case of multiple matches, and of no match. The translation $x \in X_1$ with the best match score specifies the location of M_t in I_{t+1}. We call this translation x^*.

4.2 Updating the model

Having found the best translation, x^*, of M_t with respect to I_{t+1}, the new model M_{t+1} is constructed by selecting those nonzero pixels of I_{t+1} that are within distance δ of nonzero pixels of $M_t \oplus x^*$. This is done by dilating M_t by a disk of radius δ, shifting this by x^*, and then computing the logical **and** of I_{t+1} with the dilated and translated model.

In order to allow for models that may be changing in size (for example objects stretching or getting closer or farther away), the size of the array in which M_{t+1} is stored is increased whenever there are a significant number of nonzero pixels near the boundary, and is decreased whenever there are relatively few nonzero pixels near the boundary. The height and width of the model array are adjusted separately.

The initial model, corresponding to the first frame of the image sequence, must be computed specially because there is no previous model. The user specifies a rectangle in the first frame that contains the initial model. The image is then processed to select a subset of the edge pixels in this rectangle. This is done by assuming that the camera does not move between the first two frames, and using this fact to filter the first image frame based on the second image frame (with the filtering operation described below in Subsection 5.1). Those edge pixels in the user-selected window that moved between the first and second frames are then used as the initial model. Thus the only input from the user is a rectangle in the first frame of the image sequence that contains the moving object (and as little else as possible).

5 Extensions to the basic method

In this section we describe three extensions to the basic tracking method presented above. The first of these is a filtering process, in which the stationary parts of the image frame I_{t+1} are removed before the model M_t is matched to it. This improves the performance of the method in cluttered scenes. The second extension is searching for a 'lost object'. When the tracker cannot find M_t in I_{t+1}, it tries matching previous models to the image. The third extension deals with situations where the image may contain multiple objects of similar shape. In this case, simple trajectory information is used to help disambiguate multiple possible matches.

5.1 Filtering stationary background

For cluttered scenes or highly textured images, the basic tracking method can be quite sensitive to the choice of the value of δ (where recall that δ is the threshold for determining whether or not a point of I_{t+1} is made part of M_{t+1}). The problem is that if δ is too small then the model will not tolerate much change in shape from one frame to the next. On the other hand, if δ is even moderately large, the model will start to pick up clutter in the background as part of the object. One means of dealing with this problem is to eliminate certain forms of clutter from the image and from the model. In particular, things that did not change from one frame to the next are not worth tracking, and can be removed from I_{t+1}. When the object moved but the camera stayed still, such a filtering process will remove much of the background in an image. When the camera moves, in general nearly all of the image will have changed, so the filtering process will do very little.

The filtering is implemented as follows. Any edge pixel of I_{t+1} that is directly superimposed on an edge pixel of I_t is removed from I_{t+1} as a pre-processing step (prior to matching M_t or constructing M_{t+1}). A shot-noise filter is then applied to remove small connected components (e.g., pixels in I_{t+1} that extend just beyond the structures of I_t). Note that if nothing moves from frame I_t to I_{t+1}, then the filtering will remove the entire image. This is fine, however, because if nothing moved then no tracking is required; we simply let $M_{t+1} = M_t$. This is not the same as losing the object; in that case there is change in the image from one frame to the next but the model cannot be located.

5.2 Finding a lost object

The tracking method as described thus far does not work when the object being tracked temporarily disappears from view or changes shape suddenly from one frame to the next. In such cases the pair of frames where there is a large change in image shape will result in no matching being found of the model M_t to the image I_{t+1} at the distance τ. When this happens, we compare additional models to that image frame, I_{t+1}, in which the object no longer was found. These additional models are the models from certain previous time frames, $M_{t_1}, M_{t_2}, \ldots$ $(1 \leq t_i \leq t)$ which were chosen as 'canonical' views of the object. We refer to this as 'hunt down mode', because the past history of the system is used to hunt for the missing object.

In this mode, the 'canonical' models are matched to the image, until either a good match is found, or the set of canonical models has been exhausted. The first canonical model that matches the image is then used to construct M_{t+1} in the normal manner. If none of the canonical models match I_{t+1} then the tracker has lost the object at that time frame. No model array M_{t+1} is constructed for such a frame. Rather than giving up when no match is found, the tracker continues trying to match M_t and the set of canonical models to each successive image frame until a matching image frame is found.

The set of canonical models, $\mathcal{C}$, is constructed by selecting 'distinctive-looking' models from certain frames of the image sequence. After each model M_{t+1} is created, it is compared with the current set of canonical models. If the model is sufficiently different from all the models in $\mathcal{C}$, then it constitutes a new view of the object and is added to the set. This comparison is done using the *bidirectional* minimum Hausdorff distance. The process of comparing each model with the set $\mathcal{C}$ is relatively fast, because the different models are in general approximately the same size, and thus there are not many translations to consider.

For the image sequences in this paper, and a number of others we have looked at, there are generally around 10 or 12 canonical models formed for a given object (out of the 100 frames). It should be noted that the process of matching multiple models to I_{t+1} is not much slower than just using the single model M_t. This is because many of the computations performed when matching M_t to I_{t+1} may be re-used when matching other models with I_{t+1} (see [4]).

The sequence in Figure 3 shows two people walking around, and contains several frames where one person goes behind the other. In some of these cases, the tracking method temporarily loses the object being tracked. For such frames, no model is generated. After a few frames, the object comes back into view, and it is found again (often using the canonical models to do so).

5.3 Disambiguating multiple matches

The tracking method as described thus far may not work for sequences containing multiple moving objects of approximately the same 2D shape. In such cases the model M_t may be incorrectly located in the image I_{t+1}. In this subsection we discuss some extensions to the tracker, in which simple trajectory information is used to help with this problem. (Note this also addresses the case in which multiple matches of a model are found due to erroneous matching of the model to the background, but in practice we have not observed this to happen.)

The idea is to use trajectory information in order to identify frames where the object being tracked might be incorrect. For instance, frames in which the trajectory of the object being tracked has changed suddenly are candidates for places where the tracker might have made an error and started to track the wrong thing in the image. In such a case, the tracking program enters 'suspicious mode' in which additional models are used in order to see if any of them match the image as well as M_t does. In effect, this checks whether M_t corresponds to some impostor in the image.

Currently we use very simple trajectory information, in order to detect situations in which the tracker may have started to follow the wrong object. The 'trajectory' of the object is defined to be the translation from frame I_{t-1} to I_t. That is, the model M_{t-2} matched I_{t-1} at some translation x^*_{t-1} and the model M_{t-1} matched I_t at some translation x^*_t. The vector $v = x^*_t - x^*_{t-1}$ is defined to be the current trajectory of the object. The motion of the object is simply $w = x^*_{t+1} - x^*_t$, where x^*_{t+1} is the location of the match of M_t to I_{t+1}. When there is a significant change between v and w, the tracker enters suspicious mode.

When the tracker goes into suspicious mode, it matches the models from the previous several frames (currently the previous 5 models). Given all the matches of these models and M_t, the tracker selects the match that has the closest trajectory to the previous frames. Once the tracker goes into suspicious

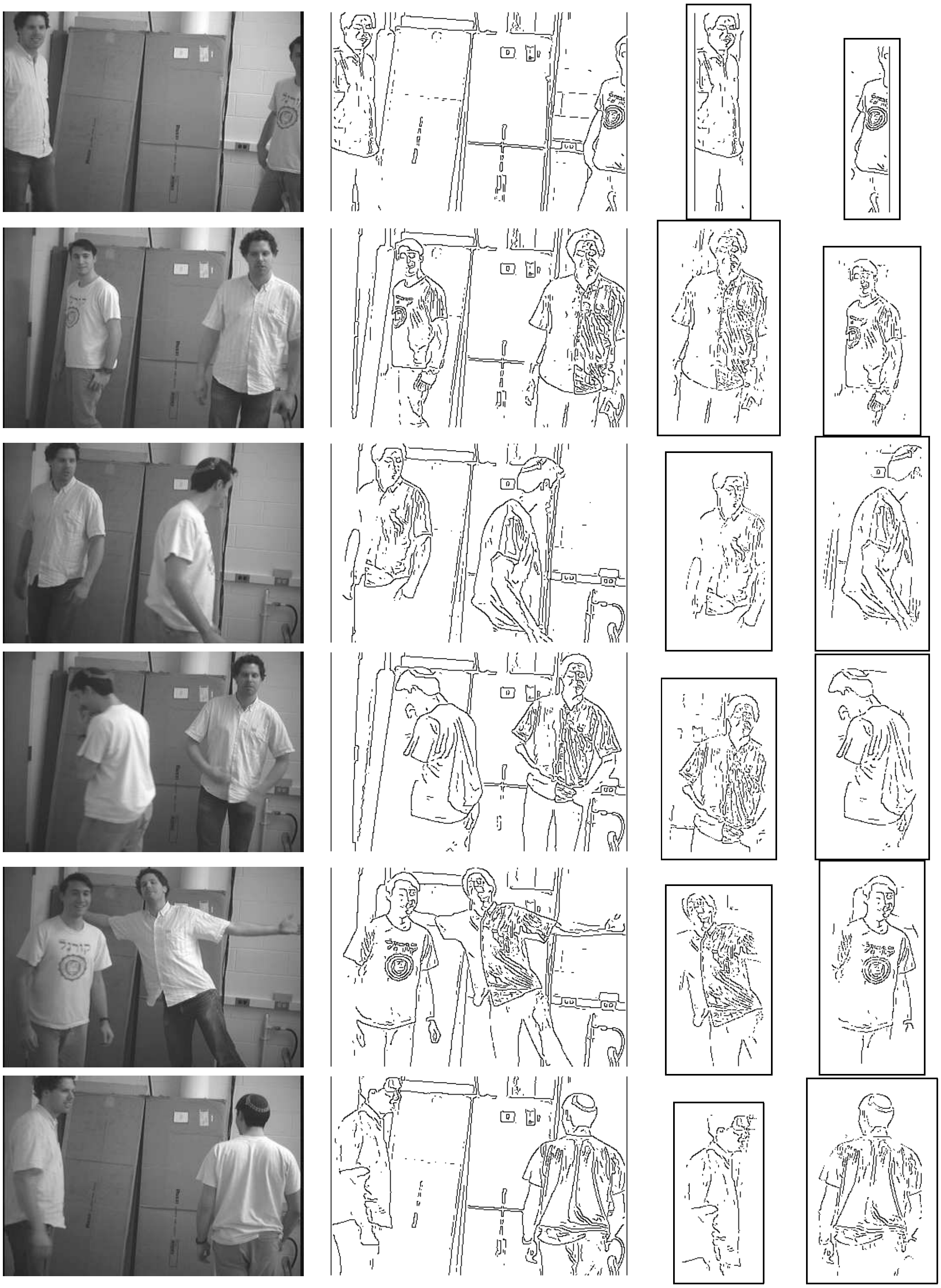

Figure 3: Six selected frames (numbers 1,20,40,60,80 and 100) from a 100 frame motion sequence, and the corresponding models of the two objects being tracked.

mode, it stays that way for several frames (currently 5 frames or until one of the other models matches better than M_t, whichever comes first). This is because the object being tracked may be temporarily hidden from view, so the tracker gives it a few frames to reappear.

Using 'suspicious mode', objects in image sequences such as the one shown in Figure 3 can be correctly tracked. During the sequence in the figure, each object passed in front of the other several times, and so was obscured from view. The objects also changed their two-dimensional shape quite significantly. Nonetheless, each one was successfully tracked through the entire sequence, and the tracker did not misidentify one object as the other (although it did temporarily track the wrong person for a frame or two while the correct person was hidden from view). This sequence illustrates the potential power of model-based methods in which two-dimensional shape is used to track an object. Without such additional global constraints (or overly strong trajectory constraints), it is very difficult to track objects that pass behind other similar objects.

6 Summary

The steps of the tracking method are thus as follows.

1. Filter the image edges of I_{t+1} to remove stationary background.
2. Find the best matching position of the model from previous time, M_t, with respect to the filtered image I_{t+1}, using the partial directed Hausdorff distance as in equation (6) with $K = \lfloor fm \rfloor$ (where we have used $f = .8$ and $\tau = 10$ pixels).
   - Trajectory information is used, as described in subsection 5.3, to disambiguate multiple matches.
   - The set of of canonical models, $\mathcal{C}$, is used to try to locate the object if no match of M_t to I_{t+1} is found.
3. Construct the new model, M_{t+1}, by selecting those pixels of I_{t+1} that are within distance δ of pixels of the translated M_t (where we have used $\delta = 8$ pixels). Update the size of the model array if required. Add M_{t+1} to $\mathcal{C}$ if it is different enough from all the current members of $\mathcal{C}$.

Although there are several parameters to the algorithm, we have used the same values for these parameters in processing many image sequences, including those in the paper.

In summary, the method consists solely of matching two-dimensional geometric models to two-dimensional edge images. There is no limitation on the search in the image — the model can translate anywhere from one frame to the next. The non-translational motion, however, must be small from one frame to the next, as this is considered to be a change in the two-dimensional shape of the object. The method can successfully track objects through image sequences in which they change overall shape substantially, they move large distances from one frame to the next, and they may be partially or fully occluded in several successive frames.

References

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

[2] E. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision Applications*, 1:223–240, 1988.

[3] D. Gennery. Tracking known three dimensional objects. In *Second National Conf. on Artificial Intelligence*, pages 13–17, 1982.

[4] 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.

[5] D. Koller, K. Daniilidis, T. Thórhallsson, and H.-H. Nagel. Model-based object tracking in traffic scenes. In *Proc. 2nd European Conf. on Computer Vision*, pages 437–452, 1992.

[6] D.G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.

[7] N. Ueda and K. Mase. Tracking moving contours using energy-minimizing elastic contour models. In *Proc. 2nd European Conf. on Computer Vision*, pages 453–457, 1992.

[8] G. Verghese, K. Gale, and C.R. Dyer. Real-time, parallel motion tracking of three-dimensional objects from spatiotemporal image sequences. In Kumar et. al., editor, *Parallel Algorithms for Machine Intelligence and Vision*, pages 340–359, New York, 1990. Springer-Verlag.

[9] J. Woodfill and R.D. Zabih. An algorithm for real-time tracking of non-rigid objects. In *Proc. American Association for Artificial Intelligence Conference*, 1991.