Object Tracking Using Motion Direction Detection

Shesha Shah P. S. Sastry
Department of Electrical Engineering
Indian Institute of Science,
Bangalore - 560012 INDIA
e-mail: shesha, sastry@ee.iisc.ernet.in

Abstract

We present an algorithm for tracking objects in a video sequence, based on a novel approach for motion detection. We do not estimate the velocity field. Instead we detect only the direction of motion at edge points and thus isolate sets of points which are moving coherently. We use a Hausdorff distance based matching algorithm to match point sets in local neighborhood and thus track objects in a video sequence. We show through some examples the effectiveness of the algorithm.

KEYWORDS: *Object tracking, motion detection.*

1 Introduction

Tracking a moving object in a video sequence is an important application of image sequence analysis. Tracking is useful in many problems [12, 10, 8, 7, 6], like collision avoidance, survelliance, gesture recognition, distance education with live teacher, searching sport clips, *etc.*

In most tracking applications a portion of an object of interest is identified in the first frame and we need to track its position through the sequence of images. Ideally, if the object to be tracked can be modeled well so that its presence can be inferred by detecting some feature sets in each frames we can look for objects with required features. Many generic features such as, active blobs, Kalman snakes or characteristics of object boundary are used as features [9, 11]. If the object of interest is highly articulated then features based detection would be good [11, 12, 7]. Weather or not we can identify characteristic features of an object, motion analysis is very useful for tracking. This is because if we can estimate the motion of the object of interest then we know where to look for the object in the next frame, thus saving a lot of computation in search and feature extraction. Most methods use some motion estimate techniques. The Optic Flow Equation (OFE) [1] based techniques give a good estimate of 2D velocity field. 3D motion techniques based on point correspondence can give the real motion of the object as oppose to the 2D techniques that can detect motion only relative to the camera. Using estimated velocity fields we can predict where the object would be in the next frame and then look for the appropriate features in that region in next frame. Bretzner and Lindeberg [5] show how the performance of feature trackers can be improved by building a view-based object representation consisting of qualitative relations between image structures at different scales. Blake and Isard [15] track outlines and features of objects, modeled as curves. They established a stochastic framework for tracking curves in visual clutter using a sampling algorithm. Rehg and Cham [13] have used a probabilistic multiple-hypothesis framework for tracking highly articulated objects.

Our method for object tracking is based on the idea that we do not need full velocity field for tracking objects. Here we show that information about direction of motion at the boundary points of an object is sufficient for tracking purposes. Our method works as follows. We first detect direction of motion at each of the edge points using an algorithm based on a simple cooperative network [4]. In the implementation presented here, this results in each of the edge points getting stamped with one of the eight directions of motion or a label of no motion. Thus edge points of each frame are now segregated into clusters of coherently moving points. Some of the clusters would correspond to the boundary points (and may be interior points) of the portion of object of interest. Hence, if we identify a piece of object of interest in one frame, it can be tracked across the sequence of frames using point set matching techniques. We use a simple Hausdorff metric based algorithm for matching point sets. Since our method of motion detection gives only 2D motion, the 2D shape of the object (and hence of our point sets) does change across the frames if the object is under-

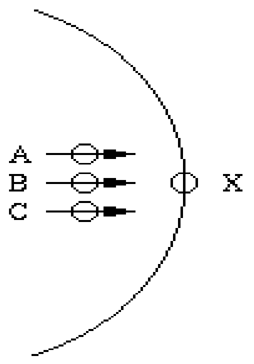


Figure 1:

going 3D motion. However across successive frames the shape change is small and our point set matching algorithm is generally robust enough for this purpose. All the same, our algorithm is best suited for the application when most of the motion of an object is in a plane perpendicular to the camera axis, e.g. distance education with live teacher, survelliance, searching sports clips, *etc*. The method as presented here uses no information other than that obtained from our motion direction detector. However any extra knowledge regarding the apparent shape of the object or any other object feature, if available, can be easily incorporated into the matching algorithm.

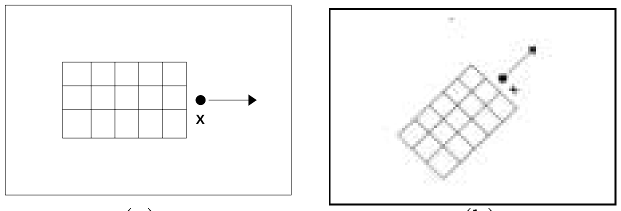
2 Object Tracking Algorithm

Our algorithm has two main parts: Detection of motion direction at edge points, matching of point sets. Each of them are explained in the next two subsections.

2.1 Motion direction detection

Our motion direction detection is a cooperative dynamic system that, at each step, updates motion direction at each point based on the detected motion at the previous step (the current state of the dynamic system) and the next image frame (the current input to the dynamic system). The intuitive idea behind such motion detection can be explained as follows.

Consider detection of motion direction at a point X shown in Fig 1. If we have detected in the previous time step that many points (shown as A, B, and C) to the left of X are moving towards X, and if X is 'a possible object point' in the current frame, then, it is reasonable to assume that part of a moving object, which is to the left of X earlier is now at X. Generalizing this idea, any point can signal motion in a given direction if it is a possible object point and if sufficient number of points 'behind' it have signalled motion in the appropriate direction earlier.



(a) (b)

Figure 2: Region of support for X (a) in $\rightarrow$ direction (b) in $\nearrow$ direction

The state of our dynamic system is represented by a binary vector with components $S_t(i,j,k)$, $1 \leq i,j \leq N; k = 1,\cdots,8; t = 1,2,\cdots$. Here the image size is NxN and t denotes time or frame number in the image sequence. $S_t(i,j,k) = 1$ represents that at time t, the point (i,j) has motion in direction k ($S_t(i,j,k) = 0, \forall k$ corresponds no motion). Given $S_t(\ldots)$, the current state of the dynamic system, and the next image frame, our algorithm updates S_t into motion S_{t+1}. As explained in the previous paragraph the updation is based on evaluation of support for motion in an oriented non-symmetric neighborhood around the point (i,j). The shape of neighborhood is shown for two different motion directions (two different values of k) in Figure 2. Given below is the equation to update S_t. The updation goes in two steps. We first calculate the support for motion in different directions, denoted by $S'_{t+1}(i,j,k)$ and then threshold it to obtain $S_{t+1}(i,j,k)$.

$$\begin{aligned} S'_{t+1}(i,j,k) = \quad & f(A \cdot \textstyle\sum_{(m,n)\in\mathcal{N}_k(i,j)} (S_t(m,n,k) \\ & \cdot FB(m,n,k)) + B \cdot \mathcal{I}(i,j) \\ & -C \cdot \mathcal{L}_k(i,j) - \tau) \end{aligned} \tag{1}$$

where,
$S_t(i,j,k)$ is the motion detector output for direction k at (i,j) at time instant t.
(i,j) is coordinate of a point and k is a direction of motion.
A is called Lateral Excitation Weight, $A > 0$
B is called Edge Excitation Weight, $B > 0$
τ is a threshold, $\tau > 0$

$$\mathcal{I}(i,j) = \begin{cases} 1 & \text{if } (i,j) \text{ is an edge point} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{L}_k(i,j) = \begin{cases} 1 & \text{if, locally, edge at } (i,j) \text{ is in} \\ & \text{direction } k \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{N}_k(i,j)$ is the excitatory neighborhood (as illustrated in Fig.2) at (i,j) in the direction k, and

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

$FB(i,j,k)$ is a 'feedback' signal determined as follows,

```
if (S'_{t+1}(i,j,k_max) − 1/7 · Σ_{l≠k_max} S'_{t+1}(i,j,l))
        > (δ · S'_{t+1}(i,j,k_max))
then
        FB(i,j,k_max) = 1 and FB(i,j,l) = 0, ∀ l ≠ k_max
else
        FB(i,j,k) = 1 ∀ k
```

$$ \tag{2}$$

where $k_{max} = arg \max_l \; S'_{t+1}(i,j,l)$

We binarize $S'_{t+1}(i,j,k)$ to obtain $S_{t+1}(i,j,k)$ *i.e.*

$$S_{t+1}(i,j,k) = \phi\left(S'_{t+1}(i,j,k)\right) \tag{3}$$

$$\phi(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{4}$$

The constants A, B, τ in eq(1), the size of neighborhood N_k and the parameter δ in calculation of FB are the parameters of the algorithm.

The intuition behind motion updation as specified by eq (1) is as follows. The value of S' is determined by essentially three terms, two of which are positive and one is negative. The first term, in the argument of f in eq (1), counts the amount of support in an oriented neighborhood for motion at (i,j) in direction k, given the motion at the previous time. The value of A, the lateral excitatory weight, determines the amount of weightage to be give to this support. In this term we sum over the neighborhood N_k, $S_t(m,n,k) \cdot FB(m,n,k)$. The feedback signal $FB(m,n,k)$ decides whether or not point (m,n), which is in the neighborhood around point (i,j), should support motion at (i,j) in direction k. From eq (2), it can be seen that, generally, the FB signal assures that each point (m,n) support motion in only one direction at all (i,j). This is needed because oriented neighborhood around a point overlap and hence it is possible to get positive support for many of such directions of motion at each point. The feedback essentially segregates such multiple evidence so that the same evidence is not compared twice, while gathering support. From eq (2), it is easy to see that the calculation of FB signal is such that if there is a predominant motion at a point then such point will support only one motion direction. Till enough evidence is built up at a point the point is potentially considered to be capable of supporting motion in all directions. There are two more terms in eq (1) for gathering support for a motion direction. The positive term with a coefficient B, the edge excitatory weight, is meant to give a large weight to edge points. This is because edge point can be thought of as points which are 'object points'. The term with coefficient C, line inhibition weight, is meant to discount the evidence of the edge point for motion direction k, if the local edge direction at that point is also k. The threshold τ decides the minimum amount of support needed to flag motion at a point. We need edge detection to get $I(i,j)$ and $\mathcal{L}(i,j)$. Any edge detector can be used. We have used non-directional Canny edge detector [2] and the edge directions are calculated locally.

We also need proper initialization for this dynamic system. Initialization is done in two cases, To start the algorithm, at $t = 0$, we need to initialize motion. To get $S_0(i,j,k) \forall i,j,k$, we run one iteration of Horn-Schunk OFE algorithm [1] at every point. We also need to initialize motion when a new moving object comes into frame for the first time. This can potentially happen at any time. Hence, in our current implementation, at every instant, we run one iteration of OFE at a point if there is no motion in a 5x5 neighborhood of the point.

More details of this motion detection algorithm can be found in [4]. The idea of using feedback as a mean of segregating evidence, which is useful in many early vision processing, is discussed in [3].

2.2 Hausdorff Distance based Image Comparison

Our motion direction detection algorithm segregates edge points into clusters of coherently moving points. The portion of the object of interest in two frames would then correspond to such two clusters in consecutive frames. So, to track an object we need a method of matching such point sets. We use Hausdorff distance for this. We need to compare a motion points set with motion points of a sub-image in the direction of object motion in the next frame. Since we assume that object does not move drastically, we search sub-images in a local neighborhood to find object in next frame.

Given two sets of points $A = \{a_i\}, i = 1, \cdots, n$ and $B = \{b_j\}, j = 1, \cdots, m$, the Hausdorff distance [14] is defined as

$$H(A,B) = \max\left(h(A,B), h(B,A)\right) \tag{5}$$

where,

$$h(A,B) = \max_{a\in A}\min_{b\in B} ||a-b|| \qquad (6)$$

The function h(A,B) is called the directed Hausdorff 'distance' from A to B. Function $h(.,.)$ is not symmetric and thus is not a true distance. It identifies the point that is farthest from any point of B, and measures the distance from A to its nearest neighbor in B. Thus the Hausdorff distance, H(A,B), measures the degree of mismatch between two sets, as it reflects the distance of the point of A that is farthest from any point of B and vice versa. Intuitively, if the Hausdorff distance is d, then every point of A must be within a distance d of some point of B and vice versa. Primarily the motion of a moving object will be translation in consecutive frames. Every iteration, based on current motion direction we take a few sub-images (of motion points), $X_k, k = 1, \cdots, K$, from next frame. The Hausdorff distance is measured between the motion points in X_k and object O. Also note that Hausdorff distance computation differs from many other shape comparison methods as no correspondence between the points in the two sets is attempted. The complete pseudo code for tracking is given below.

```
1. Let $t = 1$.

2. Select an object, $O$, from frame, $F_t$.
   Run Motion detector on $F_t$. Find the
   direction of motion for object $O$ (that
   is, majority direction of motion of
   moving points in $O$). Let $O_m$ denote the
   point set of moving points in $O$.

3. Set $t = t + 1$. Get next frame $F_t$ and run
   motion direction detector.

4. for ( sub images, $X^k$ in motion direction
   of $O$ )
   $\hat{k} = \arg\min_k H(X^k, O_m)$

5. Update object, $O \leftarrow X^{\hat{k}}$ and direction of
   motion for object $O$.

6. Go to 3.
```

3 Simulation

In this section, we present some simulation results of object tracking on two video sequences. Note that the tracking is done without actually estimating dense motion field. For the simulations we have taken $A = 100$, $B = 800$, $C = 500$, $\tau = 1000$ and $\delta = 0.7$ in our motion detection algorithm.

Figures 3 and 4 give the details of the results. The square marked in Fig.3 (a) is the object selected. Fig.3 (a)–(f) show image frames (at time $t = 3, 6, 10, 14, 25, 35$) in a video sequence where two men are walking towards each other and the result of tracking is marked by a rectangle. In Fig 3(d), at time $t = 14$, we see that the two men are overlapping. Since our model separates motion directions correctly, we are able to track object even during occlusion like this. Fig.4 gives tracking results for another video sequence where we track tail of an airplane flying up.

4 Conclusion

In this paper we presented a tracking algorithm based on a novel idea of motion direction detection. This motion detector gives global coherent motion. The results of tracking just based on motion direction are good and fast. Such tracking is very useful for various application like survelliance, distance education with live teacher, searching sport clips *etc.* We do not need to estimate velocity vector at each lattice point to track an object. In some application (*e.g.* interception), if the exact value of velocity is required then it could also be locally calculated.

References

[1] B. K. P. Horn and B. G. Schunck, **Determining optic flow**, Artificial Intelligence , V 17, 1981, pp. 185–203

[2] John Canny, **A Computational Approach to Edge Detection**, IEEE Trans. Pattern Anal. Machine Intell. Vol PAMI-8, **6** (1986) pp. 679–698.

[3] P. S. Sastry, S. Shah, S. Singh, K. P. Unnikrishnan, **Role of Feedback in Mammalian Vision: A New Hypothesis and a Computational Model**, Vision Research, Vol 1, pp 131-148, Jan 1999.

[4] Prashanth Nayak K., Shesha Shah, P. S. Sastry, **A Feedback Based Algorithm for Motion Detection**, Communication Control and Signal Processing-2000, organised by IEEE seciton, Bangalore, INDIA, Jul 2000.

[5] Lars Bretzner, and Tony Lindeberg, **Qualitative Multi-Scale Feature Hierarchies for Object Tracking** Technical report ISRN KTH/NA/P–99/09–SE

[6] Blake, A., Bascle, B., Isard, M., and MacCormick, J., **Statistical models of visual shape and motion**. Phil. Trans. R. Soc. Lond., A(356), 1283-1302,(1998).

[7] Bregler, C., **Learning and recognizing human dynamics in video sequences**, In Computer Vision and Pattern Recognition (pp. 568-574), 1997.

[8] Cedras, C. and Shah, M. , **Motion-based recognition: A survey**, Image and Vision Computing, 13(2), 129-155, (1995).

[9] Dickinson, J., Jasiobedzki, P., Olofsson, G., and Christensen, H., **Qualitative tracking of 3-d objects using active contour networks**, In Computer Vision and Pattern Recognition (pp. 812-817),(1994).

[10] Duric, Z., Rivlin, E., and Rosenfeld, A. , **Understanding object motion**, Image and Vision Computing, 16(11), 785-797,(1998).

[11] Heisele, B., Kreel, U., and Ritter W. , **Tracking non-rigid, moving objects based on color cluster flow**, In Computer Vision and Pattern Recognition (pp. 257-260), (1997).

[12] Gavrila, D. and Davis, L. **3-d model-based tracking of humans in action : a multi view approach**, In Computer Vision and Pattern Recognition (pp. 73-80). (1996)

[13] T.J. Cham and J.M. Rehg. **A multiple hypothesis approach to figure tracking**, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, volume II, pages 239-245, Fort Collins, CO, 1999

[14] Daniel P. Huttenlocher and William J. Rucklidge , **A Multi-Resolution Technique for Comparing Images Using the Hausdorff Distance** . Technical Report :cstrl.cornell/TR92-1321, December 1992

[15] Blake,
A. and Isard, M. **Condensation - conditional density propagation for visual tracking**, IJCV 1998. **ftp://ftp.robots.ox.ac.uk/pub /ox.papers/VisualDynamics/ijcv98.ps.gz**

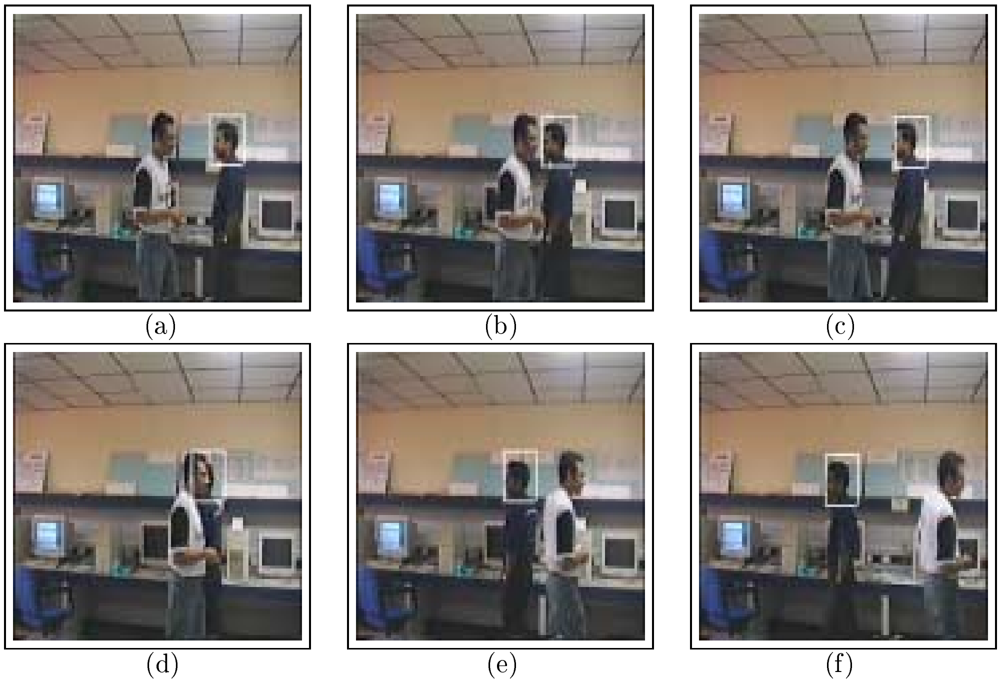


Figure 3: 2 men walk sequence (a) at $t = 3$, tracking head of a man marked inside a box. (b) at $t = 6$ (c) at $t = 10$ (d) at $t = 14$ (e) at $t = 25$ after occlusion (f) at $t = 35$

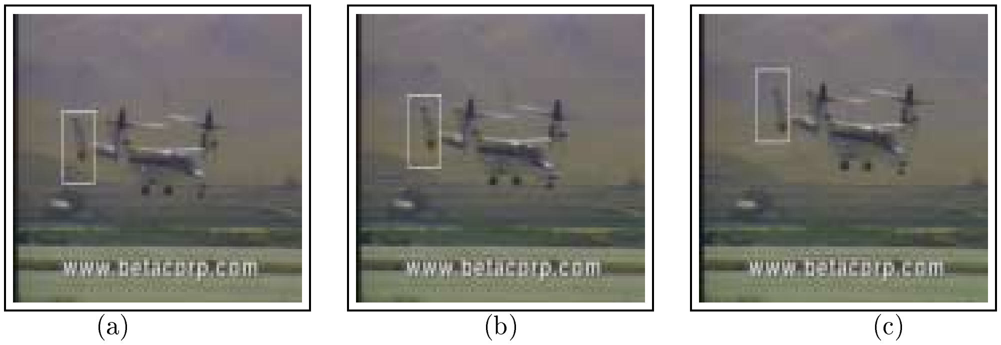


Figure 4: Air plane sequence (a) at $t = 3$, tracking tail of a plane marked inside a box. (b) at $t = 10$ (c) at $t = 20$