CVGIP: IMAGE UNDERSTANDING
Vol. 54, No. 1, July, pp. 56–73, 1991

Establishing Motion Correspondence*

KRISHNAN RANGARAJAN AND MUBARAK SHAH

Computer Science Department, University of Central Florida, Orlando, Florida 32816

Received May 5, 1989; accepted December 12, 1989

Given n frames taken at different time instants and m points in each frame, the problem of motion correspondence is to map a point in one frame to another point in the next frame such that no two points map onto the same point. This problem is combinatorially explosive; one needs to introduce constraints to limit the search space. We propose a *proximal uniformity* constraint to solve the correspondence problem. According to this constraint, most objects in the real world follow smooth paths and cover a small distance in a small time. Therefore, given a location of a point in a frame, its location in the next frame lies in the *proximity* of its previous location. Further, resulting trajectories are smooth and uniform and do not show abrupt changes in velocity vector over time. An efficient, non-iterative polynomial time approximation algorithm which minimizes the *proximal uniformity* cost function and establishes correspondence over n frames is proposed. It is argued that any method using smoothness of motion alone cannot operate correctly without assuming correct *initial correspondence*, the correspondence in the first two frames. Therefore, we propose the use of gradient based *optical flow* for establishing the initial correspondence. This way the proposed approach combines the qualities of the gradient and token based methos for motion correspondence. The algorithm is then extended to take care of restricted cases of occlusion. A metric called *distortion measure* for measuring the goodness of solution to this n frame correspondence problem is also proposed. The experimental results for real and synthetic sequences are presented to support our claims. © 1991 Academic Press, Inc.

1. INTRODUCTION

Given n frames taken at different time instants and m points in each frame, our problem is to come up with a correspondence of a point in one frame to another point in the next frame such that no two points map onto the same point (see Fig. 1). The correspondence is required in many computer vision algorithms, and there exist a number of approaches to this problem. A slightly simplified form of this problem is encountered in *stereo*, where the tokens from the left image are to be matched to those in the right image to compute the disparity. The disparity is proportional to the depth of the objects the tokens belong to in the three dimensional world. The problem here is simplified due to the fact that the possible matches can only occur along an *epipolar* line. In the case when the objects are moving and frames are taken at different time intervals, the possible matches can occur anywhere in the next frame, unlike stereo where the frames are separated in space not in time.

Motion correspondence in two frames can be used to compute optical flow and structure from motion. If the correspondence is known for a number of successive frames, one can generate a path followed by a point lying on an object. A path can be generated by starting from a point in the first frame and ending at some point in the last frame, touching each frame at not more than one point, and by joining a point in a frame by a straight line with its corresponding point in the next frame. Such a path we call a *trajectory*. Each trajectory is identified by a point in the first frame. A set of non-intersecting paths which together involve all points in all frames is a *trajectory set*. A trajectory set can be analyzed to identify important *events* in the motion of the object. The discontinuities in speed, direction, and acceleration are good candidates for the *events*. This is an alternate approach for utilizing the motion characteristics of objects without actually recovering the structuure [1].

The correspondence problem is combinatorially explosive. For instance, with n frames and m points in each frame, the number of possible trajectory sets is $m!^{(n-1)}$. To give an idea, the number of possible trajectory sets for $m = 5$, $n = 4$ is given as $(5!)^3 = 120^3 = 1{,}728{,}000$. There are two interesting points to be noted here. First, trying all possible trajectory sets will be almost impossible even for a moderate number of frames and points. Second, even if we know all possible trajectories, how do we determine which set is the correct one?

In order to cope with the complexity of this problem, the researchers have used a number of constraints. The constraints include *maximum velocity, small velocity change* or *smoothness of motion, common motion, consistent match, rigidity,* etc. [2]. The maximum velocity constraint implies that if the bound on the velocity is

* The research reported here was supported by DOD/PMTRADE under contract number N61339-88-G-0002/6. The content of the information herein does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

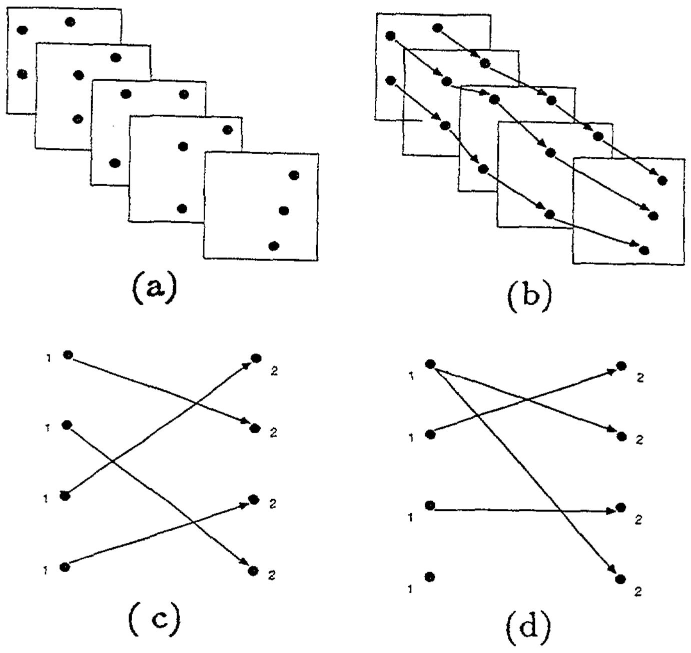


FIG. 1. (a) Input to the correspondence algorithm. Filled circles denote feature points, and squares denote image frames. (b) Output of the correspondence method, a set of trajectories. Each trajectory has exactly one point in each frame and each point in a frame belongs to exactly one trajectory, hence the mapping is isomorphic. (c) Valid correspondence. (d) Invalid correspondence.

known a priori, given a position of a point in one frame one can limit the search for possible match in the next frame to a small neighborhood of the position in the present frame. The small velocity change heuristic assumes that the direction and speed of the motion cannot change by a large amount, so that one can eliminate some false matches. This constraint essentially leads to *smooth* motion [3]. Common motion constraints the motion of the points in a small neighborhood to be similar, and consistent match forces only one match for one point.

The rigidity and smoothness of motion assumptions have received the most attention. Rigidity implies that the objects in the three dimensional world are rigid, therefore the Euclidean distance between any two points on the rigid object will remain unchanged in the next frame. In fact, Ullman and Yuille [4] have attempted to show that smoothness follows from rigidity; that is, if the objects are rigid their underlying motion will be smooth, but not necessarily vice versa.

An important issue in the correspondence problem is to convert the above *qualitative* heuristics into quantitative expressions, which become the cost functions. Then the aim is to search for the trajectory set which minimizes one of these functions. Enumeration of all possible sets and picking the one with the least cost is not possible. Therefore, one needs to use a good approximation algorithm to obtain a sub-optimal solution that is very close to the optimum solution.

We propose the *proximal uniformity* (see Fig. 2) constraint to solve the correspondence problem. According to this constraint, most objects in the real world follow a smooth path and cover a small distance in a small time. Therefore, given the location of a point in a frame, its location in the next frame lies in the *proximity* of its previous location. Further, resulting trajectories are smooth and uniform and do not show abrupt changes in velocity vector over time.

We have also designed an approximate algorithm which runs in polynomial time and gives a reasonably good solution, but not necessarily the optimum one. We observe that it is meaningful to use an approximate algorithm for this problem if and only if two trajectories having similar cost resemble each other in space; and that for a cost function based on smoothness of motion to be effective, the initial correspondence should be assumed correct. In our method the initial correspondence is determined by the use of the optical flow method proposed by Little *et al.* [5]. The proposed algorithm is also able to deal with cases of limited occlusion.

The next section briefly reviews related work in motion correspondence. The formulation of the problem is stated in Section three. An exact expression for the proximal uniformity function and justification for its use are also described in Section three, while the algorithm which operates on the proposed cost function and establishes correspondence is outlined in Section four. Section five deals with the description of the distortion measure. Results are presented in Section six, and the discussion of the results is given in Section seven. Finally the use of optical flow in obtaining initial correspondence is discussed in Section eight, and a modified algorithm to deal

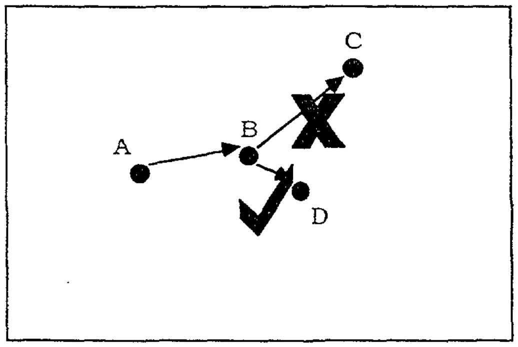


FIG. 2. Proximal uniformity. Trajectories ABC and ABD show the same amount of smoothness in velocity, but trajectory ABD is more proximally uniform than trajectory ABC.

with limited cases of occlusion is outlined in Section nine.

2. RELATED WORK

Ullman [6] has proposed a minimal mapping theory for correspondence. He formulated the correspondence as a minimization problem, the correct correspondence being the one for which the sum of the negated logarithmic values of the probabilities of point velocities is a minimum. Ullman's approach is probabilistic in nature; he assumed that each point is moving independent of every other point. When the points being tracked belong to the same object, they move as a rigid structure and violate the independence assumption. But, when the points in a frame belong to different moving objects, the independence assumption is better obeyed, as points on different objects move independently of each other. At low velocity the cost function used by Ullman reduces to the distance. Though he proposes the use of correspondence for recovering structure based on rigidity, the rigidity assumption is not used explicitly in establishing correspondence.

Jenkin [7] presented a method for tracking the three dimensional motion of points from their changing two dimensional perspective images as viewed by a nonconvergent binocular vision system. The scheme used the concept of velocity smoothness. Given the initial 3-D positions and velocity of the points and a sequence of frames, the scheme tracked the position and velocity of points in 3-D. At any instant of time it considered two frames. A frame is composed of a left stereo image and a right stereo image. The stereo correspondence and the velocity in the first frame are known and the stereo correspondence and the velocity in the next frame are to be determined. It should be noted that the position of a point in 3-D can be obtained from the stereo correspondence. He used a greedy strategy for choosing the best solution.

Aggarwal, Davis, and Martin [8] have proposed two different methods to solve correspondence: iconic methods and structure based methods. Iconic methods compare segments from two different frames by computing normalized cross correlations of pixel values, absolute differences of pixel values, and squares of differences in pixel values. These methods, though computationally costly, lend themselves to fast parallel implementations. The structure based method consists of identifying the interesting points in the two frames and the matching problem is tackled through the Hough transform. This algorithm is able to tackle shape and size deformation to some extent and also occlusion.

Bernard and Thompson [9] use an iterative algorithm to match feature points selected in two different frames taken in a small time interval. This algorithm starts with initial probabilities for matches between pairs of points based on a *common motion* heuristic by correlating a small neighborhood around the feature points, and restricts the potential match for a point using the heuristic that a point in the world does not move a large distance between frames. The probabilities are refined to strengthen common motion of neighboring points. The algorithm terminates when all the probabilities of matches between pairs of point are close either to 1 or to 0.

Sethi and Jain [3] have proposed two different iterative algorithms, GE (Greedy Exchange) and MGE (Modified Greedy Exchange), which minimize the cost function called *path coherence*. GE assumes the initial correspondence between frame 1 and 2 to be known. Then it extends the trajectories frame by frame, from frame 3 to frame n. When it is trying to extend the trajectories to frame k, it assumes the correspondence to be the nearest match to start with. It then goes into an iterative loop, exchanging correspondences which improves the cost function value. When no exchange is favorable, it repeats the process for frame $k + 1$. This process continues until correspondences in frame n are established. The difference between GE (Greedy Exchange) and MGE (Modified Greedy Exchange) is that GE assumes an initial correspondence between the first and second frames based on the nearest neighbor criterion, which does not change till the end, while in MGE, the initial correspondence is not assumed, and the above process is repeated in forward and backward directions, which could alter the initially assumed correspondence.

Williams and Hanson [10] have formulated the correspondence of line tokens between two views as finding a minimal arc cover in a weighted bipartite graph. The detected line tokens in the two views are the nodes in the graph and the cost on each arc is the error measure. Their formulation accomodates the split and fusion of the tokens also. They make use of the optical flow measurements computed by Anandan's method [11] to predict the location in the second view of a line token L from the first view and consider all line tokens in a small rectangular window around the predicted location as potential matches for L. The error measure for correspondence of L with a potential match is its average distance with the potential match. They further outline a procedure for computing depth based on the length of two corresponding line segments.

Recently, Weng, Ahuja, and Huang [12] have proposed a method to match two views of a scene. They minimize a matching function which has terms for difference in intensity, edgeness, and cornerness and terms which encourage a pixel to have similar motion to its neighbors. The terms of edgeness and cornerness are again functions of intensity, and it would have been better computationally if the number of terms capturing the same information had been reduced. Their algorithm is iterative. The method has been tried on matching two different views of the same static scene.

3. FORMULATION OF THE PROBLEM

We are given a sequence of n frames denoted by $f^1, f^2, \ldots, f^n$. We assume that the important tokens in each frame have already been identified using a corner detector [13] or an interest operator. Therefore, each frame f^i is a set of points. Corresponding to the ith point in the jth frame, we have its 2-D coordinates denoted by a vector X_i^j. Our aim is to come up with a one to one onto correspondence Φ^k between points of the kth frame and the $(k + 1)$th frame.

It is not unrealistic to assume that in space objects move small distances in a small time interval, and their corresponding motion is smooth or uniform. If the time interval between frames is small, then the 2-D projection of 3-D motion will also be small and smooth. Therefore, the location of a point in the next frame will be in the *proximity* of the location in the previous frame. Smoothness of the motion implies minimum change in *velocity* of the point; that is, the object cannot change its direction and speed instantaneously. Hence the objects will follow a *proximal uniform path*. We propose to establish correspondence by minimizing the *proximal uniformity function* δ, which will prefer the proximal uniform path,

$$\delta(X_p^{k-1}, X_q^k, X_r^{k+1}) = \frac{\|\overline{X_p^{k-1}X_q^k} - \overline{X_q^kX_r^{k+1}}\|}{\sum_{x=1}^m \sum_{z=1}^m \|\overline{X_x^{k-1}X_{\Phi^{k-1}(x)}^k} - \overline{X_{\Phi^{k-1}(x)}^kX_z^{k+1}}\|} + \frac{\|\overline{X_q^kX_r^{k+1}}\|}{\sum_{x=1}^m \sum_{z=1}^m \|\overline{X_{\Phi^{k-1}(x)}^kX_z^{k+1}}\|},$$

where $1 \leq p, q, r \leq m$; $2 \leq k \leq m - 1$; $q = \Phi^{k-1}(p)$; $\overline{X_q^kX_r^{k+1}}$ is the vector from point q in frame k to the point r in frame $k + 1$; and $\|X\|$ denotes the magnitude of the vector X.

Figure 3 shows a 3 frame 3 point sequence. The correspondence between frames 1 and 2 is known. The correspondence between points in frame 2 and those in frame 3 is to be established. The computation of the proximal uniformity function $\delta(X_1^1, X_1^2, X_1^3)$ in this case is, as an example,

$$\delta(X_1^1, X_1^2, X_1^3) = \frac{\|\overline{X_1^1X_1^2} - \overline{X_1^2X_1^3}\|}{C1} + \frac{\|\overline{X_1^2X_1^3}\|}{C2},$$

where

$$\begin{aligned} C1 = {} & \|\overline{X_1^1X_1^2} - \overline{X_1^2X_1^3}\| + \|\overline{X_1^1X_1^2} - \overline{X_1^2X_2^3}\| \\ & + \|\overline{X_1^1X_1^2} - \overline{X_1^2X_3^3}\| + \|\overline{X_2^1X_2^2} - \overline{X_2^2X_1^3}\| \\ & + \|\overline{X_2^1X_2^2} - \overline{X_2^2X_2^3}\| + \|\overline{X_2^1X_2^2} - \overline{X_2^2X_3^3}\| \\ & + \|\overline{X_3^1X_3^2} - \overline{X_3^2X_1^3}\| + \|\overline{X_3^1X_3^2} - \overline{X_3^2X_2^3}\| \\ & + \|\overline{X_3^1X_3^2} - \overline{X_3^2X_3^3}\| \\ C2 = {} & \|\overline{X_1^2X_1^3}| + |\overline{X_1^2X_2^3}\| + \|\overline{X_1^2X_3^3}\| \\ & + \|\overline{X_2^2X_1^3}| + |\overline{X_2^2X_2^3}\| + \|\overline{X_2^2X_3^3}\| \\ & + \|\overline{X_3^2X_1^3}| + |\overline{X_3^2X_2^3}\| + \|\overline{X_3^2X_3^3}\|. \end{aligned}$$

The proximal uniformity function obeys the following criteria.

- Speed does not change much between two successive frames.
- Direction does not change much between two successive frames.
- Displacement of a point between two successive frames tends to be small.

In our proximal uniformity function the first term represents a *relative* change in velocity, while the second term denotes a relative displacement. The second term forces the proximal matches, while the first term leads to smooth and uniform trajectories. Note that the numerator in each term represents an *absolute* quantity; for example, the numerator of the first term represents an absolute change in velocity of a point q in frame k. The

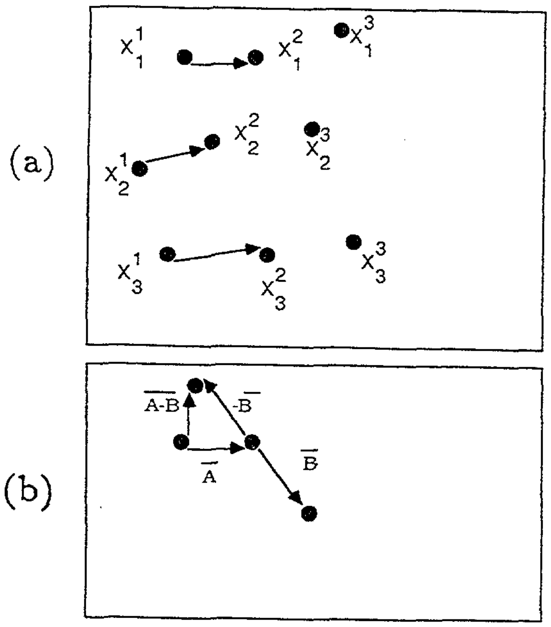


FIG. 3. (a) 3 frames with 3 points in each frame. (b) $\overline{A}$ denotes $\overline{X_1^1X_1^2}$. $\overline{B}$ denotes $\overline{X_1^2X_2^3}$. $\overline{A - B}$ denotes $\overline{X_1^2X_2^3} - \overline{X_1^1X_1^2}$.

denominators denote sums of absolute quantities for all possible matches. Hence, the ratio gives the relative measure of quantities. Since a change in velocity is a vector quantity, the magnitude in the first term incorporates the change in both *speed* and *direction*.

In our formulation we assume that Φ^1, an initial correspondence, is known. Φ^k is determined such that $\sum \delta(X_p^{k-1}, X_q^k, X_r^{k+1})$ is minimized. We believe that for a smoothness based method to be meaningful, the initial correspondence should be known. Given the correct initial correspondence, the algorithm will correctly grow the trajectories. Applying the smoothness constraint alone without knowing any details about the initial movement of the points may in many cases lead to false trajectory sets.

Previously, researchers have missed one or the other important point in the cost function to be minimized. For instance, Ullman [6] asserted that humans prefer motion which minimizes the *distance* as objective function. He noted that his *minimal mapping* requires only the measurement of distances between elements, not of the directions, a property that might have an advantage in terms of economical implementation. Ullman also showed that most of the time resulting trajectories are non-intersecting. On the other hand, Sethi and Jain [14] first proposed that only *direction* should be considered in the cost function for minimization. Later in [3] they used the direction as well as a speed term. Most of the time their function prefers intersecting trajectories. Our proximal uniformity function is biased neither to intersecting nor to non-intersecting trajectories. Using our notations the cost function Ψ used by Sethi and Jain is

$$\begin{aligned}&\Psi(X_p^{k-1}, X_1^k, X_r^{k+1})\\&\quad = 0.1 * \left(1 - \frac{\|\overline{X_p^{k-1}X_q^k}\| \cdot \|\overline{X_q^kX_r^{k+1}}\|}{\|\overline{X_p^{k-1}X_q^k}\| * \|\overline{X_q^kX_r^{k+1}}\|}\right)\\&\qquad + 0.9 * \left(\left(1 - \frac{2 * [\|X_p^{k-1}X_q^k\| * \|X_q^kX_r^{k+1}\|]^{1/2}]}{\|X_p^{k-1}X_q^k\| + \|\overline{X_q^kX_r^{k+1}}\|}\right),\right.\end{aligned}$$

where the first term captures smoothness in direction and the second term smoothness in speed. We have noted that when their cost function is minimized, the resulting trajectories tend to prefer matches between points which are far apart. This is due to the fact that the second term in the above function can be reduced to a term having the displacement in the denominator, and change in speed in the numerator. Hence, for the same change in speed, this function gives a lower value for correspondences involving greater displacement than for correspondences involving smaller displacement.

Figure 4(a) shows this anomaly. In all our figures points have been labeled with the frame number to which they belong. Figure 4(a) violates the basic assumption that between successive frames points do not move a great deal, which has been used in many previous approaches. For instance, Ullman [6] has established this point in which he derives the probability distribution curve of 2D velocity in image plane to be a monotonically decreasing curve, even though the 3D velocity distribution curve may not be so in the lower velocity ranges. This implies that near matches are more probable than distant matches. Figure 4(b) shows the expected correspondence and is preferred by our method.

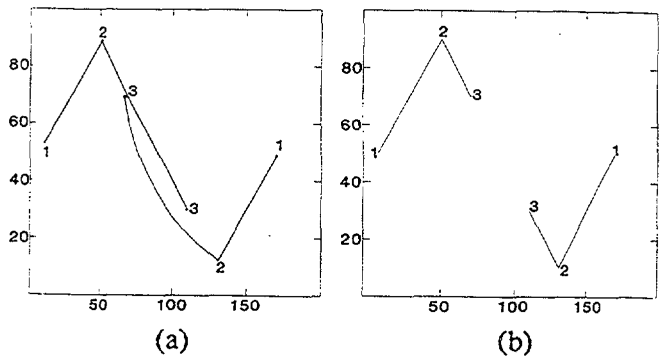


FIG. 4. Anomaly. (a) Wrong correspondence preferring longer distance. (b) Correct correspondence.

4. ALGORITHM

We have designed a non-iterative greedy algorithm **A** which assigns correspondence of the points in one frame with the points in the next frame, keeping the overall proximal smoothness function as close to the minimum as possible in addition to being fair to each individual assignment.

Algorithm A

```
1. For $k = 2$ to $n - 1$ do
   (a) Construct M an $(m * m)$ matrix, with the point from $k$th frame along the rows and points from $(k + 1)$th frame along the columns.
   (b) Let $M[i, j] = \delta(X_p^{k-1}X_i^kX_j^{k+1})$, when $\Phi^{k-1}(p) = i$.
   (c) for $a = 1$ to $m$ do
       i. Identify the minimum element $[i, l_i]$ in each row $i$ of $M$.
       ii. Compute *priority matrix* $B$, such that $B[i, l_i] = \sum_{j=1, j \neq l_i}^{m} M[i, j] + \sum_{k=1, k \neq i}^{m} M[k, l_i]$ for each $i$.
       iii. Select $[i, l_i]$ pair with highest *priority* value $B[i, l_i]$, and make $\Phi^k(i) = l_i$.
       iv. Mask row $i$ and column $l_i$ from $M$.
```

In our algorithm, when we consider the minimum along the rows, it could happen that more than one minimum lies along the same column j. That is, more than one point in frame k competes for point j in the $(k + 1)$th frame. To get a one to one onto mapping, we should choose only one of these. However, our scheme should not just choose the minimum possible combination quantitatively, but should prefer a combination where each individual correspondence is fairly good. The correspondence from frame k to $k + 1$ involves m points. The minimum correspondence could be very favorable to some $(m - 1)$ points and not favorable for the mth point. We should prefer a correspondence which is equally favorable to all points; at the same time we should not end up with a very high proximity path uniformity function. The algorithm was designed to take care of these conditions.

The rationale behind the priority measure is that if $[i, j]$ is not assigned, any other element along the ith row or jth column can get assigned. Assuming they are all equally probable, their average value is a good measure for selecting the order of assignments. We should choose the one with the highest priority measure and assign that first. Priority measure is a rough estimate of the alternate assignment to $[i, j]$.

This algorithm has the nice property that it will pick the *least cost assignment* if there are just two points in the frame. Consider the matrix M, with $M[1, 1] = 0.6$, $M[1, 2] = 0.3$, $M[2, 1] = 0.7$, and $M[2, 2] = 0.2$. The minimum along row 1 is element $[1, 2]$ with value 0.3, while the minimum along row 2 is element $[2, 2]$ with value 0.2. Therefore, $B[1, 2] = (0.6 + 0.2) = 0.8$, and $B[2, 2] = (0.7 + 0.3) = 1.0$. Now, $B[2, 2] > B[1, 2]$, hence we choose correspondence (2, 2) first. Then, mask row 2 and column 1 with a high value. Next we pick the only assignment possible, $[1, 1]$. For this assignment $\delta = M[1, 1] + M[2, 2] = 0.6 + 0.2 = 0.8$, which is the least possible for this configuration.

Let us take a closer look at the priority measure. A low priority measure means the other values along row i and column j are very good candidates as well. Hence, there is more competition for assignment $[i, j]$. By choosing that assignment which has the highest priority measure, we are committing to an assignment which we know will give a very large value assignment if we miss. It should be noted that this heuristic tries to keep the worst assignment as small as possible. Since we are considering only the minimum possible values along each row we are keeping the proximal path uniformity function value as low as possible. Also, since we blank out the ith row and jth column after assigning $[i\, j]$, we ensure one to one onto mapping between points in frames k and $k + 1$. Hence, we can say that our algorithm gets a one to one onto assignment which is close to the optimum assignment. And it also has the property that no individuual assignment will be very bad.

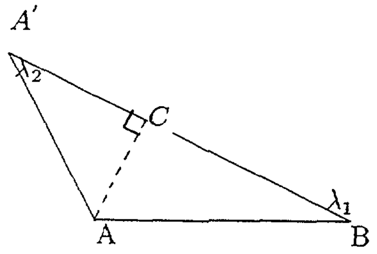
It should be noted that in an n frame correspondence problem, with minimum proximal path uniformity the assignments in two consecutive frames need not be the minimum possible. An algorithm which extends the trajectory by one frame at a time starts with an initial correspondence between frames 1 and 2 and extends it to 3 and then to 4 up to n. In that case, getting the optimum assignment between the frames k and $k + 1$ need not necessarily lead to a globally minimum trajectory set. In that sense our heuristic assignment between frames k and $k + 1$, in addition to making a one to one onto assignment and keeping the proximity path uniformity value close to minimum, also keeps the worst individual assignment low. Thus it is suited to n frame correspondence even though it looks at only three frames at a time.

5. PERFORMANCE MEASURE FOR CORRESPONDENCE

Every sub-optimal algorithm produces an approximate solution. In many applications, we need to know how good a given approximation is. Therefore, we propose a measure called the *distortion measure* for grading the determined trajectories. The higher the distortion measure, the greater is the displacement of the *established* trajectory from the *actual* trajectory. The distortion of a trajectory is computed as follows. Take the square of Euclidean distance between a point as it appears in the actual trajectory and its position in the established trajectory. Do this for each frame and add the distortion in all frames to get the distortion measure of a trajectory. Add the distortion measures of all trajectories in the sequence of frames to get the distortion measures of a trajectory set. By the nature of the problem, it is more reasonable to use the summation of distortions of a *trajectory set* as a performance measure, rather than distortion of any individual trajectory. By the definition of the distortion measure, an ideal cost function should have the following properties:

- The minimum trajectory set should have zero distortion.
- The higher the cost function value, the greater should be the distortion measure.

Now, we will justify the use of Euclidean distance as distortion measure. Let A and B be the projections of two points lying on a rigid object. Assume A was identified as A' due to the error in the correspondence process. Many approaches to structure from motion consider the square of the image distance between two points as the rigidity term. Hence, the error passed on to the structure recovering algorithm is denoted by $|A'B|^2 - |AB|^2$. We show that $|AA'|^2$, the square of the Euclidean distance between the actual location (A) of a point and its distorted location (A'), is a good measure of the error.



$$|A'B| = |A'C| + |CB|$$
$$= |AA'| \cos \lambda_2 + |AB| \cos \lambda_1$$
$$|A'B|^2 - |AB|^2 = |AA'|^2 \cos \lambda_2^2 + |AB|^2 \cos \lambda_1^2$$
$$+ 2|AB|\,|AA'| \cos \lambda_1 \cos \lambda_2 - |AB|^2$$
$$= |AA'|^2 \cos \lambda_2^2 - |AB|^2 \sin \lambda_1^2$$
$$+ 2|AB|\,|AA'| \cos \lambda_1 \cos \lambda_2.$$

But

$$|AC| = |AA'| \sin \lambda_2 = |AB| \sin \lambda_1$$
$$|A'B|^2 - |AB|^2 = |AA'|^2 \cos^2 \lambda_2 - |AA'|^2 \sin^2 \lambda_2$$
$$+ 2|AA'|\,|AB| \cos \lambda_1 \cos \lambda_2$$
$$= |AA'|^2[\cos^2 \lambda_2 - \sin^2 \lambda_2]$$
$$+ 2|AB|\,|AA'| \cos \lambda_1 \cos \lambda_2.$$

Since $|AB|^2$ is the rigidity term for the link between A and B, the error in rigidity from distorting A with A' is ($|A'B|^2 - |AB|^2$). In the expression derived above we could assume that $|AB|$ is a constant as the object is rigid, and we could ignore the effects of cosine and sine functions in comparison with $|AA'|^2$, $|AA'|$, and $|AB|$, as they vary from +1 to −1. Hence, the dominant factor in this expression is $|AA'|^2$, and it is a good measure of the error introduced by the correspondence process.

6. RESULTS

In this section we present the results for real and synthetic sequences using our algorithm. Figure 5(a) shows a synthetic set of five points in four frames used by Sethi and Jain [3]. Figure 5(b) shows the correspondence established by our algorithm. This sequence is a good example of trajectories which cross over. Next, a six frame *Blocks Sequence* obtained by digitizing two moving cardboard cutouts was considered. One of the cutouts is a triangle, and the other one is a five sided irregular polygon. Figure 6(a–b) shows the first two frames taken from that *Blocks Sequence* and Fig. 6(c) shows the last frame in that sequence. The locations of the corner points picked manually were input to our method and the resultant trajectories are shown in Fig. 6(d). To see if our method was sensitive to the number of objects in the scene, we separated the feature points of the triangle and the irregular polygon and presented them to our method. The correct set of trajectories was generated in both the cases. Figure 6(e) shows the generated trajectories with the points of the triangle alone input to the method and Fig. 6(f) shows the generated trajectories with the points of the irregular polygon alone input to the method.

Figure 7 shows the trajectories obtained for the *Walker Sequence*. This sequence was obtained by a program reported by Cutting [15]. In this sequence a person is

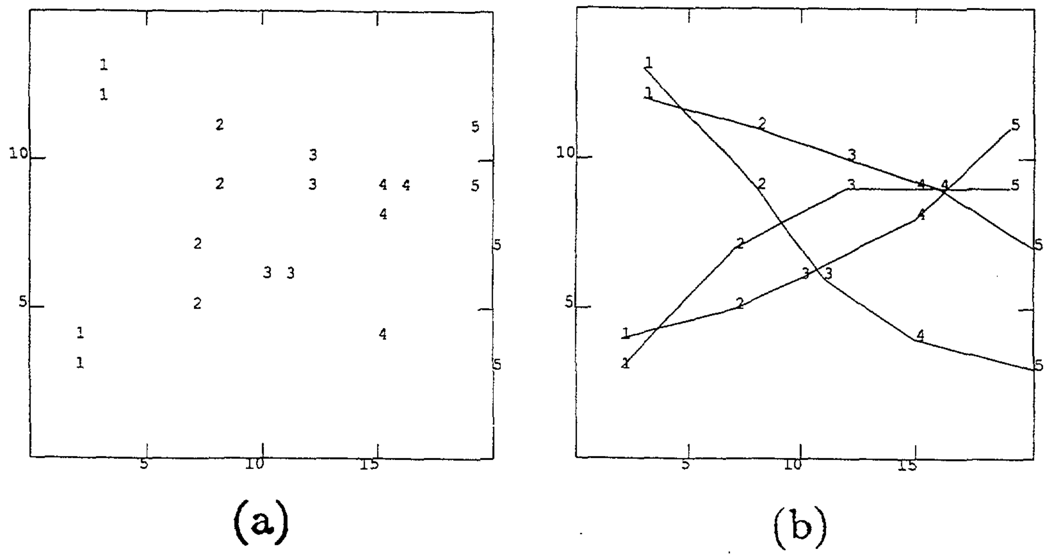
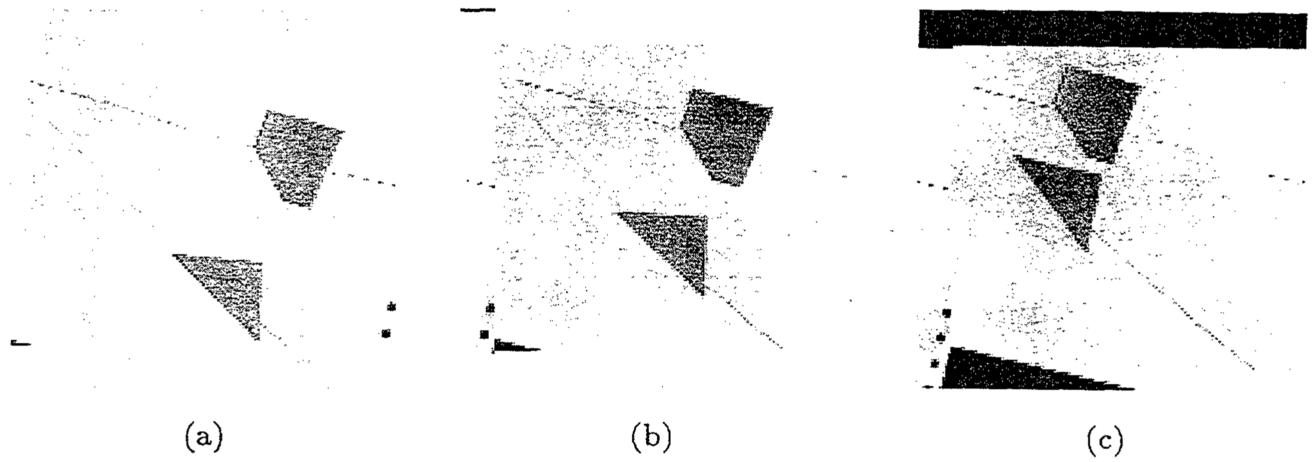


FIG. 5. *Synthetic Sequence*. (a) Four points in five frames. (b) Resultant trajectories.



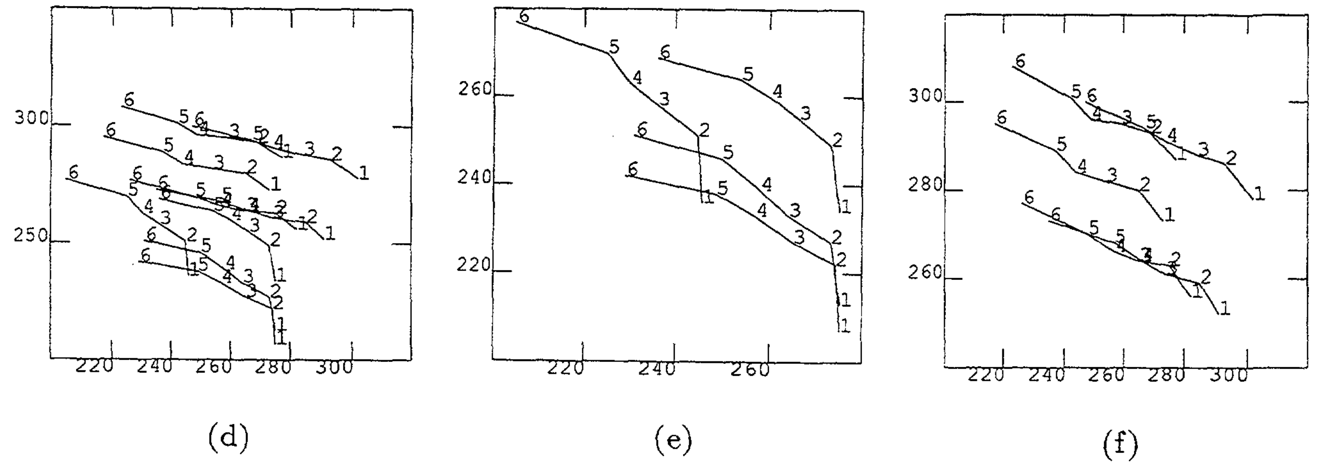
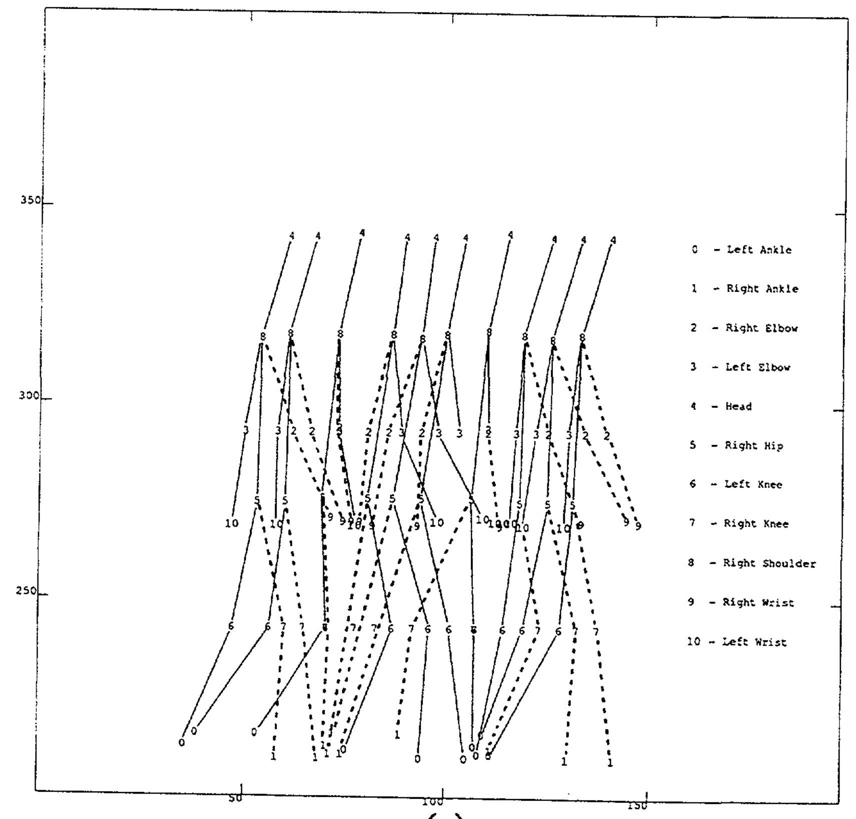


FIG. 6. *Blocks Sequence.* (a) Frame 1. (b) Frame 2. (c) Frame 5. (d) Established trajectories with corners of the triangle and the five sided irregular polygon as the input. (e) Established trajectories with corners of the triangle alone as the input. (f) Established trajectories with corners of the five sided irregular polygon alone as the input.

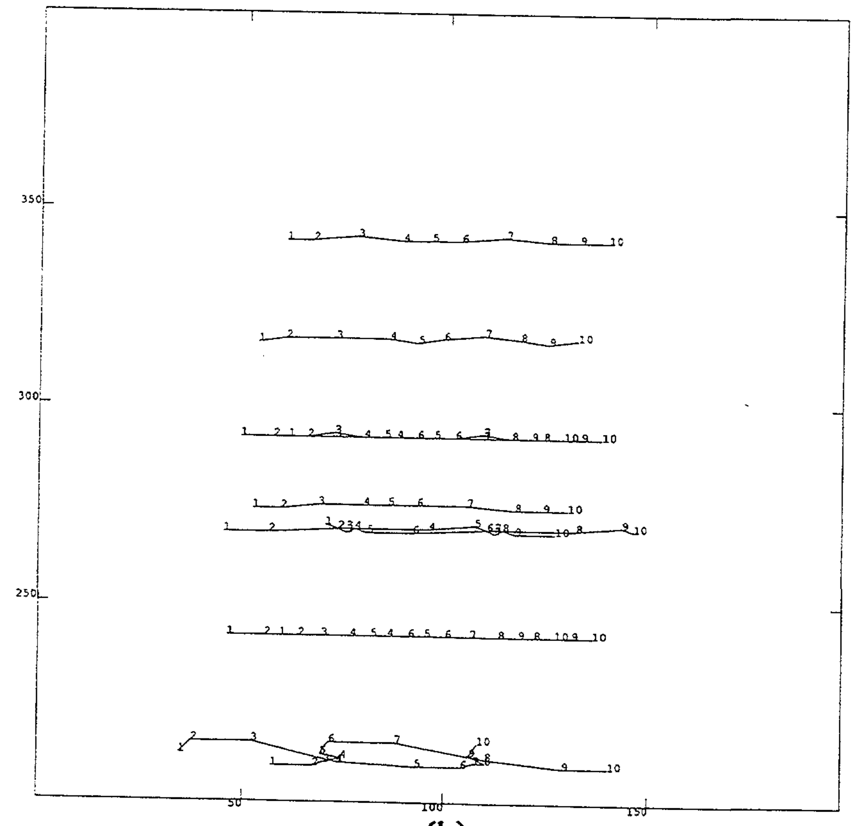
shown walking. Eleven points, head, right shoulder, left and right knee, left and right elbow, right hip, left and right wrist, and left and right ankle are tracked through 10 frames. This figure is an exception to the labeling convention we have adopted. Here the point label corresponds to a specific part of the body and the index is shown on the top right corner of the figure. The entire body moves up and down in a slight bouncing manner. The movements of the shoulder and hip are ellipsoidal and the movement of arms and legs are pendular. Finally, the results for the *Superman Sequence* are shown in Fig. 8. Six points on the heads and belts of three soldiers moving in different directions were tracked from the movie *Superman* as reported by Sethi and Jain. Actual coordinates of points in various frames are shown in Figure 8(a).

7. DISCUSSION

In order to study the effect of initial correspondence and the effectiveness of our method we conducted experiments on the 5 frame synthetic sequence having 4 points in each of the frames shown in Fig. 5(a). We applied the Sethi–Jain method to it. We tried all possible combinations, and found that the set of trajectories corresponding to the global minimum found by *exhaustive search* was totally different from the computed set of trajectories. Also, we found that these two sets of trajectories have costs close to each other and still are very different in space. But, this problem did not surface when a correct initial correspondence was assumed to be known. Figure 9(a) shows the minimum trajectory set without assuming initial correspondence. Figure 9(b) shows the trajectory set that was found by the Sethi–Jain method. Figure 9(c) shows the trajectory set with the least cost function value when the correct initial correspondence was assumed. Trajectory sets in Figs. 9(a) and (b) differ in their initial correspondence, whereas trajectories in Figs. 9(b) and (c) have the same initial correspondence. It should be noted that the trajectory set in Fig. 9(b) does not resemble the one in Fig. 9(a) even though they have similar cost, whereas the trajectory sets in Figs. 9(b) and (c) have similar costs and resemble each other more closely.



(a)

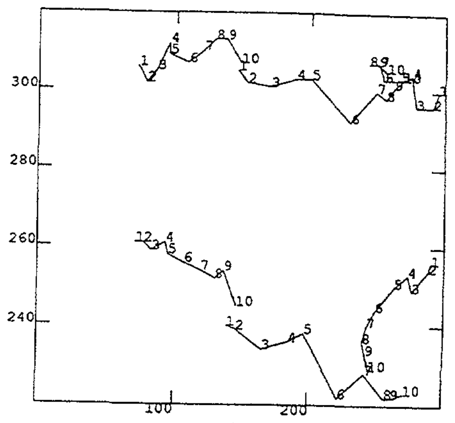


(b)

FIG. 7. (a) *Walker Sequence*. (b) Established trajectories.

frame	head-1	belt-1	head-2	belt-2	head-3	belt-3
1	(73,306)	(72,261)	(147,305)	(140,240)	(295,300)	(290,256)
2	(79,302)	(78,261)	(154,302)	(147,239)	(291,296)	(288,254)
3	(86,305)	(83,259)	(171,301)	(166,234)	(278,296)	(276,249)
4	(95,312)	(93,261)	(189,303)	(185,236)	(275,304)	(273,253)
5	(95,309)	(95,258)	(200,303)	(196,238)	(266,303)	(263,250)
6	(108,307)	(106,256)	(229,292)	(222,222)	(254,303)	(249,244)
7	(120,310)	(119,254)	(249,300)	(242,228)	(251,307)	(243,240)
8	(129,313)	(130,252)	(256,298)	(257,222)	(243,307)	(240,236)
9	(138,313)	(137,254)	(262,301)	(262,222)	(250,307)	(242,233)
10	(149,307)	(147,245)	(268,303)	(271,223)	(256,305)	(245,229)

(a)



(b)

FIG. 8. *Superman Sequence.* (a) Coordinates of points in various frames. (b) Established trajectories.

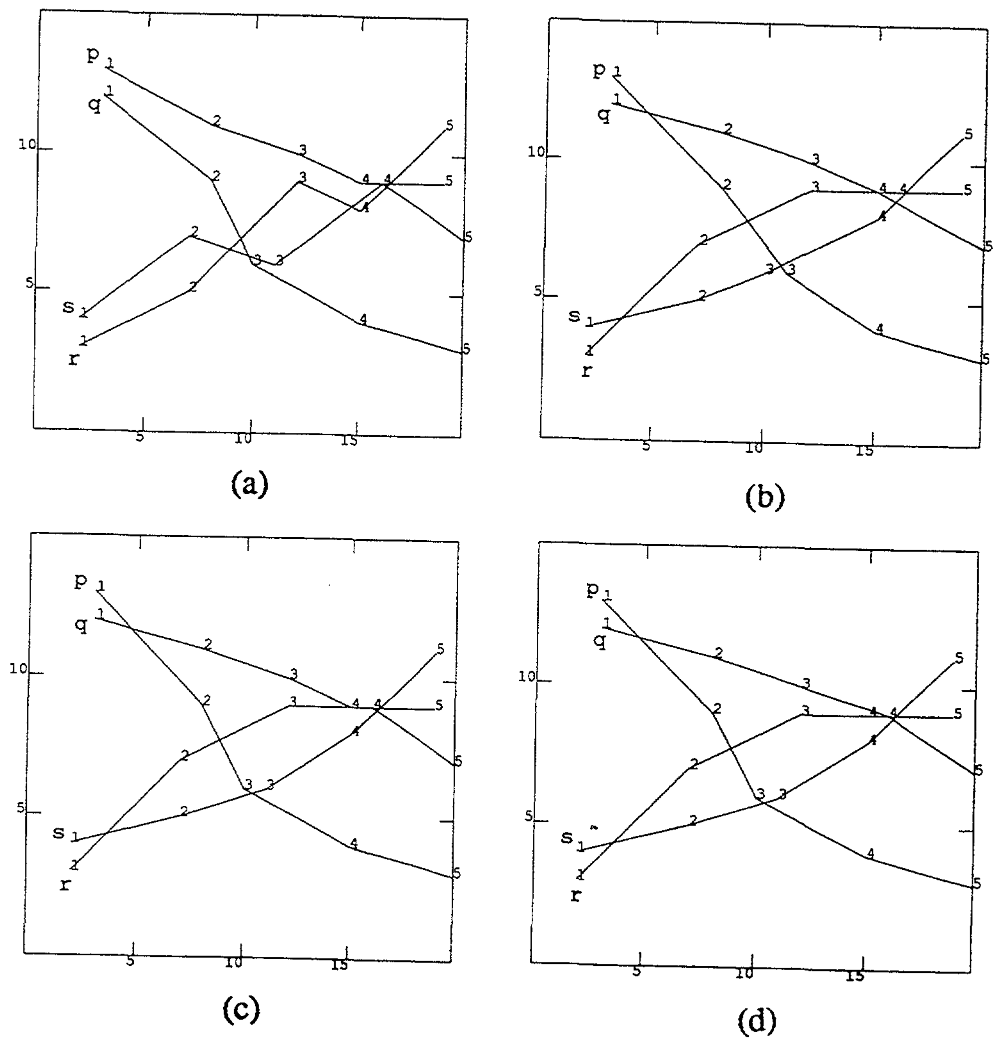


FIG. 9. Effect of initial correspondence on spatial similarity. (a) Minimum trajectory set for Sethi–Jain path coherence function without initial correspondence. (b) Trajectory set identified by Sethi–Jain method (with initial correspondence). (c) Minimum trajectory set for Sethi–Jain path coherence function with initial correspondence assumed. (d) Minimum trajectory set with proposed method. Trajectory sets in (a) and (b) have similar values on the Sethi–Jain path coherence function; however, *they are very different in space*. When trajectory sets agree on initial correspondence, similarity in space is better captured by the path coherence function. (b) and (c) have similar values of the path coherence function and they do closely resemble each other in space.

Hence, we concluded that *applying the smoothness constraint alone without knowing any detail about the initial movement of the points may in many cases lead to false trajectory sets.*

It should be noted that there are two parts to a method:

- A cost function.
- An algorithm which works using the cost function.

An exponential algorithm that gives the best result is the brute force strategy which enumerates all trajectory sets and finds the minimum trajectory set. It can be used to analyze how good an algorithm is. Figure 9(d) shows the minimum trajectory set found using our cost function. The distortion measure between the computed trajectory set and the minimum trajectory set by our method was found to be 2 as shown in Fig. 10(a). The same measure for the Sethi–Jain method was found to be 218, as shown in Figure 10(b) when initial correspondence was not assumed. The same measure for the Sethi–Jain method was found to be 12, as shown in Figure 10(c), when initial correspondence was assumed.

The above example highlights an important property any method should have. Trajectories with similar costs should resemble each other; they should not be very different in space. The distortion measure introduced in this paper essentially computes this, by computing the square of Euclidean distance. This point should be emphasized more when the search space for this problem is very large, and it is not possible to use an exhaustive search algorithm. All computationally feasible algorithms find sub-optimal solutions for this problem. It is clear that a sub-optimal solution is not meaningful if two trajectory sets with similar costs have a very high distortion measure.

It is in this context that an algorithm which starts with some arbitrary initial correspondence and later refines it to get the trajectory set with least cost may not be doing the right job. What such an algorithm is missing is the fact that two trajectory sets over the same set of points and frames may have very similar cost and still their layout in space may be totally different. It is this spatial distance that the distortion measure is capturing. It is here that the importance of correct initial correspondence comes into play. Given the correct initial correspondence velocity smoothness based schemes will grow these initial correspondences.

As shown in the sample computation in Fig. 10, focusing on the distortion value of the trajectory sets we find that our method performs better than the Sethi–Jain method. From Figure 10(d), assuming initial velocity of points (in our formulation, assuming initial correspondence) is more reasonable for schemes which use smoothness of motion for correspondence.

Also, our algorithm is a non-iterative one and is computationally faster than the Sethi–Jain method. Our algorithm always terminates and it does not get caught in local minima. When we consider the level of interaction between frames we observe that in our scheme the ith frame can affect correspondences in frames $i + 1$ to n, which is the same as that of GE. But in MGE because of the additional backward pass frame i can affect correspondences in frames $(i - 1)$ down to 1 in addition to frames $i + 1$ to n. However, the price paid is very heavy: there is a good chance of its going into an infinite loop with the forward pass undoing what the backward pass did and vice versa. Also it is not clear if this increased level of interaction is buying anything more in the end result.

Trajectory	frame-1	frame-2	frame-3	frame-4	frame-5	distortion of trajectory
p	0	0	1	0	0	1
q	0	0	0	0	0	0
r	0	0	0	0	0	0
s	0	0	1	0	0	1
Total						2

(a)

Trajectory	frame-1	frame-2	frame-3	frame-4	frame-5	distortion of trajectory
p	0	4	17	25	37	93
q	0	4	20	25	16	65
r	0	1	17	2	4	24
s	0	4	13	2	17	36
Total						218

(b)

Trajectory	frame-1	frame-2	frame-3	frame-4	frame-5	distortion of trajectory
p	0	0	1	0	0	1
q	0	0	0	0	5	5
r	0	0	0	0	5	5
s	0	0	1	0	0	1
Total						12

(c)

	With Initial Correspondence	Without Initial Correspondence
Minimum cost	0.150548	0.146734
Found cost	0.169654	0.169654
Distortion	12	218

(d)

FIG. 10. Distortion measures. (a) With proposed method. (b) Distortion of established trajectory set by Sethi–Jain method without initial correspondence. (c) Distortion of established trajectory set by Sethi–Jain method with initial correspondence. (d) Comparison table.

8. INITIAL CORRESPONDENCE USING OPTICAL FLOW

Assumption of correct initial correspondence is rather unrealistic. However, we can remove this unrealistic assumption by making use of optical flow to give an estimate of the initial image velocity of the feature points, and use it for the initial correspondence.

There are two classes of algorithms for computing optical flow: *gradient based* and *feature based.* In our case, it will be appropriate to use a *gradient based* approach for computing optical flow, since that method does not require the correspondence problem to be solved. This way we will be able to combine the gradient based technique with our correspondence algorithm for estimating the motion characteristics of objects in the scene.

We consider frames f^1 and f^2 to find the *optical flow*, and generate a *fictitious* frame, f^0, with feature points

$$\langle (x_1^0, y_1^0), (x_2^0, y_2^0), (x_3^0, y_3^0), \ldots (x_n^0, y_n^0) \rangle$$

such that

$$x_i^0 = x_i^1 - u_i^1$$
$$y_i^0 = y_i^1 - v_i^1,$$

where u_i^1 is the image plane velocity in the x direction and v_i^1 is the image plane velocity in the y direction of feature point i in frame f^1. This way we know the initial correspondence between frames f^0 and f^1, therefore the proposed correspondence algorithm can be applied to frames f^1 and f^2.

In our lab the only working implementation of the optical flow algorithm available is that of the one proposed by Little *et al.* [5]. Therefore, in order to check the proposed correspondence algorithm, we used the optical flow computed by Little *et al.* A number of experiments were performed with this initial correspondence, and in all cases the correct correspondence was achieved. As an example Fig. 11(a) shows the (x, y) image coordinates of nine points in frame f^1 to f^6 of *Blocks Sequence*. Figure 11(b) shows the optical flow vector (u, v) computed by Little *et al.*'s method, and the computed coordinates for frame f^0.

Note that Little *et al.*'s algorithm is essentially feature based. However, we assume that any good gradient based optical flow algorithm will provide similar results. Therefore, as far as the performance of our correspondence algorithm is concerned it should not make any difference.

9. CORRESPONDENCE WITH OCCLUSION OF FEATURE POINTS

An object or part of the object is occluded if it does not appear in the image due to some other object surface coming between the object and the camera. When part of an object is occluded, the feature points in that part are also occluded, and are missed in that frame. The other reason for missing feature points could possibly be due to a failure to detect the feature point, even though it was present in the frame. The only source of information for finding the occlusion is the number of feature points.

The different cases of occlusion to be handled are:

1. Some points visible in frame k get occluded in frame $k + 1$; that is, $m_{k+1} < m_k$.
2. Some points occluded in frame k become visible in frame $k + 1$; that is, $m_{k+1} > m_k$.
3. Some points visible in frame k get occluded in frame $k + 1$, and some points occluded in frame k become visible in frame $k + 1$. An interesting offshoot of this is that the number of points in frame k and frame $k + 1$ may remain the same, but still some points visible in frame k could be occluded in frame $k + 1$.

We assume that there is no occlusion in the first two frames, and all feature points show up in those frames. Hence the first time an occlusion occurs, it will be a case 1 occlusion. Our algorithm detects this occlusion and fills up for the missing point in frame $k + 1$ using the correspondence in the frames $k - 1$ and k. As every time an occlusion occurs, it is detected and filled up immediately, the algorithm never encounters cases 2–3, even though the data may have such cases.

We modify algorithm **A** to take care of occlusion as follows. When $m_k > m_{k+1}$, there is occlusion between frames k and $k + 1$. In the original algorithm we look for the minimum along a row i, to identify the point in frame $k + 1$ that suits the point i of frame k the best. But when there is occlusion, not every point in frame k is going to be assigned a point in frame $k + 1$. Hence the minimum on some rows need not be considered at all, and if considered it affects the assignment of feature points badly. Whereas every point in frame $k + 1$ is to be assigned to some point in frame k. Therefore, looking at the minimum along a column j will indicate the point in frame k that suits this point j of frame $k + 1$ the best. This strat-

point	f^1	f^2	f^3	f^4	f^5	f^6
1	(246,236)	(245,251)	(236,258)	(230,263)	(225,270)	(205,277)
2	(275,234)	(273,249)	(266,255)	(260,260)	(254,264)	(236,269)
3	(275,207)	(274,222)	(265,227)	(257,233)	(249,238)	(229,242)
4	(275,213)	(273,227)	(264,233)	(258,239)	(250,246)	(231,251)
5	(277,287)	(269,293)	(259,295)	(249,296)	(242,301)	(223,308)
6	(273,273)	(265,280)	(254,282)	(244,284)	(237,289)	(217,295)
7	(282,256)	(275,263)	(265,264)	(257,266)	(248,270)	(226,277)
8	(291,252)	(285,259)	(274,261)	(266,264)	(257,268)	(235,273)
9	(302,278)	(293,286)	(284,288)	(274,291)	(267,294)	(247,300)

(a)

point	f^1	u	v	f^0
1	(246,236)	-2	14	(248,222)
2	(275,234)	-2	15	(277,219)
3	(275,207)	-2	15	(277,192)
4	(275,213)	-2	16	(277,197)
5	(277,287)	-9	7	(286,280)
6	(273,273)	-8	7	(281,267)
7	(282,256)	-8	7	(290,249)
8	(291,252)	-8	7	(299,245)
9	(302,278)	-8	8	(310,270)

(b)

FIG. 11. (a) The coordinates of feature points in *Blocks Sequence*. (b) The constructed frame f^0 and the initial *flow vectors* (u, v) computed by the method proposed by Little *et al.*

egy is better in the presence of occlusion, as every minimum along a column is meaningful and could lead to an assignment. A similar modification, to look along the columns only, is done in computing the elements of the priority matrix. The modified algorithm **B** is as follows:

Algorithm B

```
1. For $k = 2$ to $n - 1$ do
   (a) Set up $M$, and $(m_k * m_{k+1})$ matrix, with the $m_k$ points from the $k$th frame along the row and $m_{k+1}$ points from the $(k+1)$th frame along the column.
   (b) Let $M[i, j] = \delta(X_p^{k-1} X_i^k X_j^{k+1})$, when $\Phi^{k-1}(p) = i$.
   (c) if $(m_{k+1} < m_k)$ then /*This is a case of occlusion*/
       i. for $a = 1$ to $m_{k+1}$ do
          A. Identify the minimum element $[l_j, j]$ in each column $j$ of $M$.
          B. Compute priority matrix $B$, such that $B[l_j, j] = \sum_{i=1, i \neq l_j}^{m} M[i, j]$
          C. Select $[l_j, j]$ pair with highest priority value $B[l_j, j]$, and make $\Phi^k(l_j) = j$.
          D. Mask row $l_j$ and column $j$ from $M$.
       ii. identify $m_k - m_{k+1}$ points for which correspondence has not been found. For those points create new feature points in frame $k + 1$ by extrapolating the correspondence from frame $k - 1$ to frame $k$.
       iii. Set $m_{k+1} = n$.
       else    /* No Occlusion */
       i. for $a = 1$ to $m_{k+1}$ do
          A. Identify the minimum element $[i, l_i]$ in each row $i$ of $M$.
          B. Compute priority matrix $B$, such that $B[i, l_i] = \sum_{j=1, j \neq l_i}^{m} M[i, j] + \sum_{k=1, k \neq i}^{m} M[k, l_i]$ for each $i$.
          C. Select $[i, l_i]$ pair with highest priority value $B[i, l_i]$, and make $\Phi^k(i) = l_i$.
          D. Mask row $i$ and column $l_i$ from $M$.
```

The modified algorithm first identifies points in frame f^k which do not have corresponding points in the frame f^{k+1}, and creates new feature points in frame f^{k+1} corresponding to those missing points. Let us assume that a point a in frame k with coordinates (x_a^k, y_a^k) which corresponds to a point c in frame $k - 1$ with coordinates (x_c^{k-1}, y_c^{k-1}) is such a point without a corresponding point in frame f^{k+1}. A new feature point b with coordinates (x_b^{k+1}, y_b^{k+1}) is created to correspond to point a as follows:

$$x_b^{k+1} = x_a^k + (x_a^k - x_c^{k-1})$$
$$y_b^{k+1} = y_a^k + (y_a^k - y_c^{k-1}).$$

In these equations the extrapolation between the coordinates of frame $k - 1$ and frame k is computed. This extrapolation ensures smoothness in velocity, both in magnitude and in direction. Also, this creation of new feature points ensures that there are n points in frame k, which is used during the correspondence between frames k and $k + 1$.

We tried this algorithm over the *Blocks Sequence* removing the point (257, 266) in frame f^4, and the *Superman Sequence* removing the points (189, 303) in frame f^4, and (200, 303) in frame f^5. In both cases our algorithm worked well, and the occluded point was filled up suitably, and the correspondences in subsequent frames were obtained correctly. Figure 12 shows the resultant trajectories. The points marked with * show the *new* feature points created to take care of the induced occlusion.

We also tried this algorithm for a synthetic *Polygon Sequence* with two squares, one rectangle, one triangle, and one hexagon as shown in Fig. 13. The velocity vectors of these polygons and the depth are tabulated in Fig. 14. The locations of the corners in these polygons were generated for 15 frames with unit time intervals. Whenever a corner of a polygon is occluded by some other polygon in front of it, its position is not available to the correspondence algorithm. The square in the lower part of the image does not participate in any of the occlusions, and its trajectories are spatially isolated from others. Some corners of the upper square get occluded by the hexagon in frames 3, 4, and 6. These occlusions are detected correctly and the points are filled up suitably. Some corners of the triangle are also occluded by the upper square in frames 9, 10, 11, and 12. In frame 10, one of the corners of the upper square is very close to where one of the occluded corners of triangle should have been. The algorithm makes a mistake in assigning the corner of the square to the triangle and filling in a corner for the square. In frames 11 and 12 the algorithm correctly identifies the corners of the triangle to be occluded, but the filled up location is distorted as the algorithm made a mistake in frame 10. But when all the corners of the triangle become visible again in frame 13, the propagation of the error stops. Because of the error made in frame 10, one of the trajectories belonging to the triangle is not perfectly straight. Some corners of the rectangle are occluded by the triangle in frames 13, 14, and 15. The algorithm did well to detect the occlusion and to fill up the missing points. Figure 15 shows the computed trajectories of this polygon scene.

Next, we tried our algorithm on an *Apple Sequence* with four objects, a Rubik Cube, a toy apple, a toy vessel, and a toy lid. These objects were moved on a table and a sequence of five frames were taken. The camera was

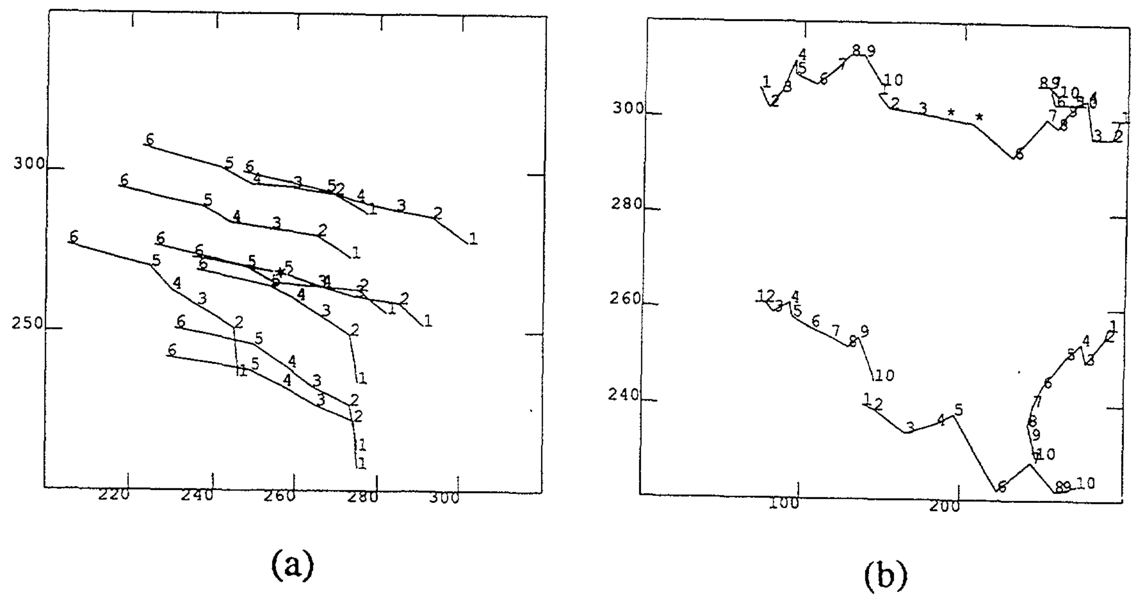


FIG. 12. (a) Trajectories of *Blocks Sequence* with a missing point in frame f^4 which is filled up by a point marked *. (b) Trajectories of *Superman Sequence* with missing points in frame f^4 and frame f^5 which are filled up with points marked *.

given a viewpoint and orientation such that the table was seen sloping in the image. If the viewpoint did not show the table sloping, the trajectories computed tended to be flat. Two points on the apple, three points on the lid, and two points on the vessel were tracked. The Rubik Cube was used to creat occlusion of the feature points. All points are visible in the first two frames. In the third frame one of the points on the cover of the vessel is hidden by the Rubik Cube. In the fourth frame the points on the apple are hidden by the Rubik Cube, one of the points on the lid is also hidden by the vessel, and another point from the lid is omitted to simulate feature points being not detected. In Frame 4, 4 points out of the 7 feature points are occluded. In frame 5, all points are visible, but one of the points on the apple is left out. Our algorithm performed well in this sequence as well detecting the occlusions properly and filling in the missing points. Figure 16(a–e) shows the sequence and Fig. 16(f) shows the trajectories. Because of poor imaging conditions the quality of the images is not very good.

Finally, this algorithm was applied to the configuration shown in Fig. 17. In this example there are two disks, one shown shaded and the other unshaded. One disk is partially occluded by the other. Trajectories for points marked 1, 2, 3, and 4 are tracked. Points 1 and 2 are on disk 1, which is shaded dark, and points 3 and 4 are on disk 2, which is unshaded. The disks are positioned such that when both the disks rotate point 1 gets occluded after 30° of rotation and Point 2 gets occluded after 120°

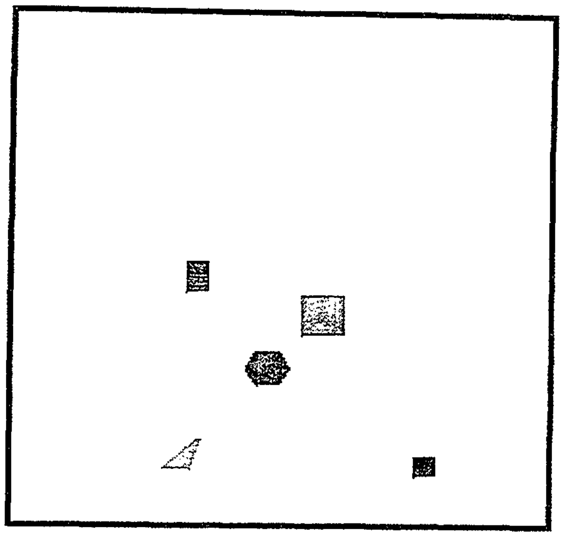


FIG. 13. First frame of the *Polygon Sequence*. There are two squares, one hexagon, one triangle, and one rectangle in the scene.

Polygon	Velocity in X direction	Velocity in Y direction	Depth
Triangle	5	5	16
Upper Square	-7	-10	15
Hexagon	10	7	10
Lower Square	-1	1	5
Rectangle	4	-8	17

FIG. 14. This table shows the velocity vectors and the depth values of the polygons. A polygon with a lower depth value occludes the polygon with higher depth value when they overlap.

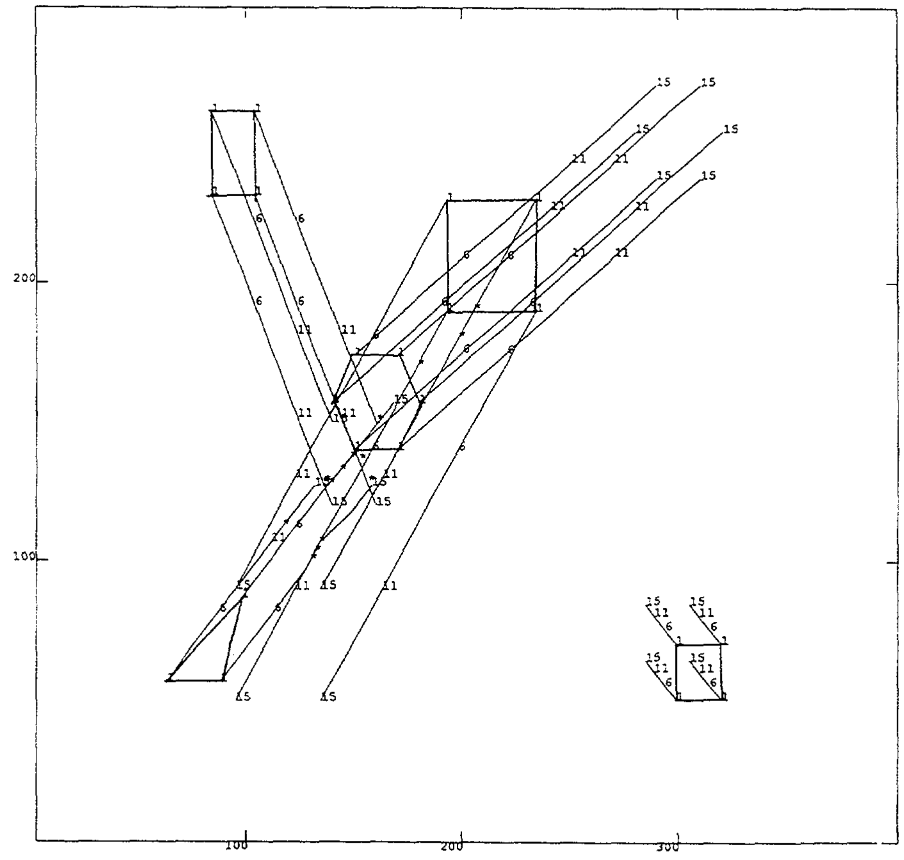


FIG. 15. The trajectories of the corners in the *Synthetic Polygon Sequence* tracked over 15 frames. The points in frames 1, 6, 11, and 15 are shown on the trajectories marked by their frame numbers. The polygons as they appear in the first frame are also drawn. Some corners of the upper square get occluded in frames 3, 4, and 6, and of the triangle in frames 9, 10, 11, and 12, while those of the rectangle get occluded in frames 13, 14, and 15. These occluded corners are detected while performing the correspondence and are suitably filled up and are shown by * in this figure.

of rotation. Point 1 remains occluded till 135° of rotation and since the points are tracked every 15° of rotation, it reappears at 150°. The instances of points when they are occluded were removed and the data were presented to our algorithm. Our algorithm could handle the occlusion of point 1 till 135°; however, it failed to give a correct correspondence when point 1 reappeared. It is due to fact that an important *event* between 105° and 135°, when the point changes its course, has been occluded. Figure 18(b) shows the results. The new feature points generated by the algorithm to take care of occlusion are marked with $*$. When we introduce the point at 135° so that the event is not occluded our algorithm gives correct results as shown in Figure 18(c).

10. CONCLUSION

We considered the problem of establishing a correspondence among n frames with each frame having m points. There are two parts to this problem. First, the introduction of constraints in order to limit the search space; this leads to a *cost function*. Second, a sub-optimal algorithm which minimizes the cost function in order to determine a set of trajectories. We proposed a *proximal uniformity constraint* which forces near matches, and with this constraint the resulting trajectories are smooth and uniform. A non-iterative algorithm was presented which gives very good results for synthetic and real sequences. In order to compare the performance of correspondence algorithms we defined a *distortion measure*, which captures the error due to incorrect correspondence. We also improved this method by making use of *optical flow* to get the *initial correspondence*. The algorithm was further modified to handle a restricted version of occlusion. As a continuation of this work, we intend to investigate the performance of our method on motion which does not obey the *smoothness of velocity* assumption.

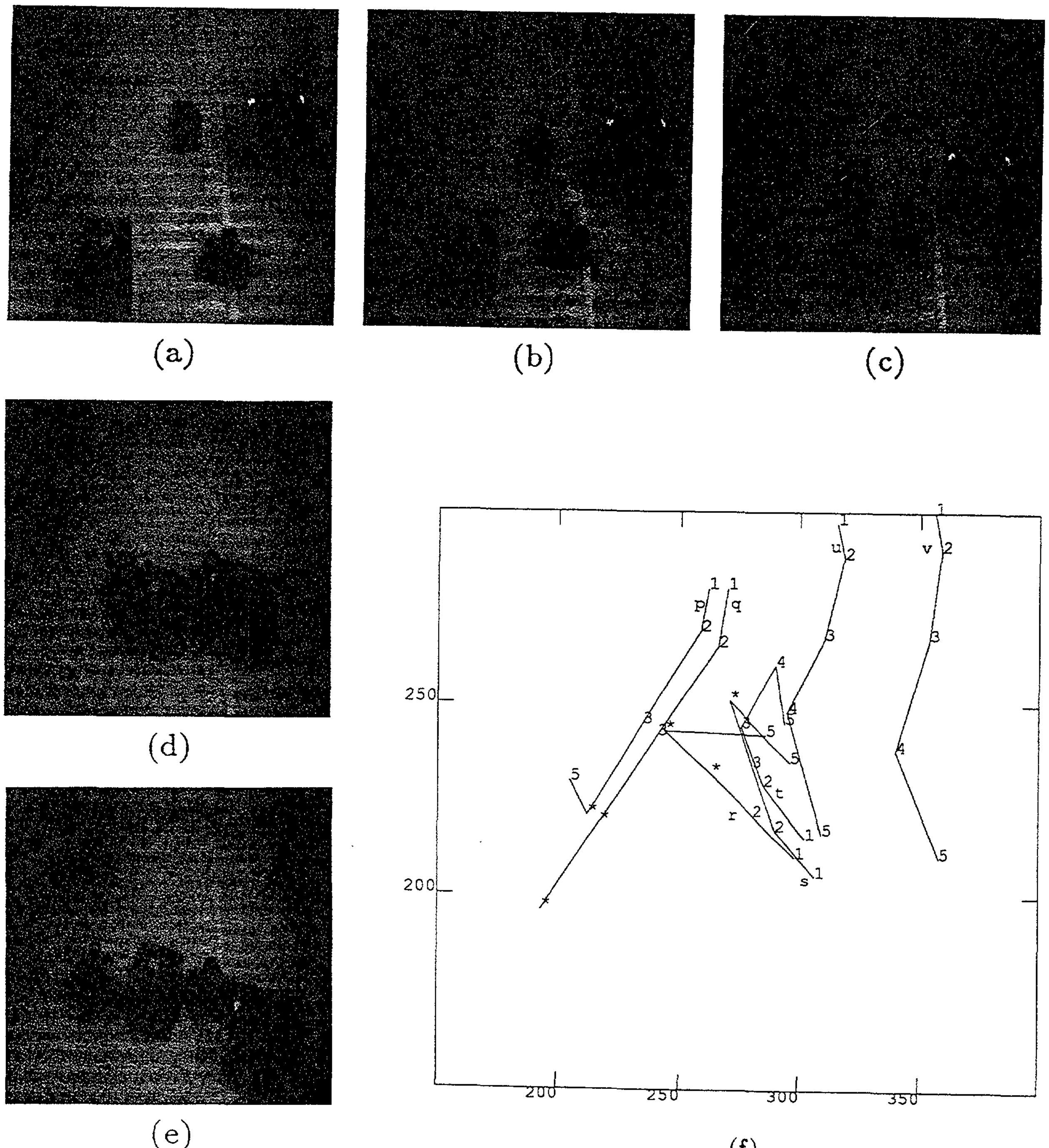


FIG. 16. A real image *Apple Sequence* with occlusion (a)–(e) frames $f^1 - f^4$. (f) Trajectories of this scene. The points in a frame are denoted by the frame number. The occluded points which are filled in are indicated by *. Trajectories p and q are of the points on the apple; r, s, and t are of points on the lid; and u and v are of the points on the vessel. No point on the Rubik Cube was tracked. It was used to occlude the feature points.

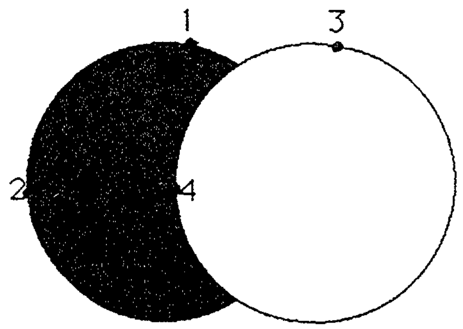


FIG. 17. Two rotating disks positioned in space such that one disk is partially occluded by the other. Trajectories for points marked 1, 2, 3, 4 are tracked.

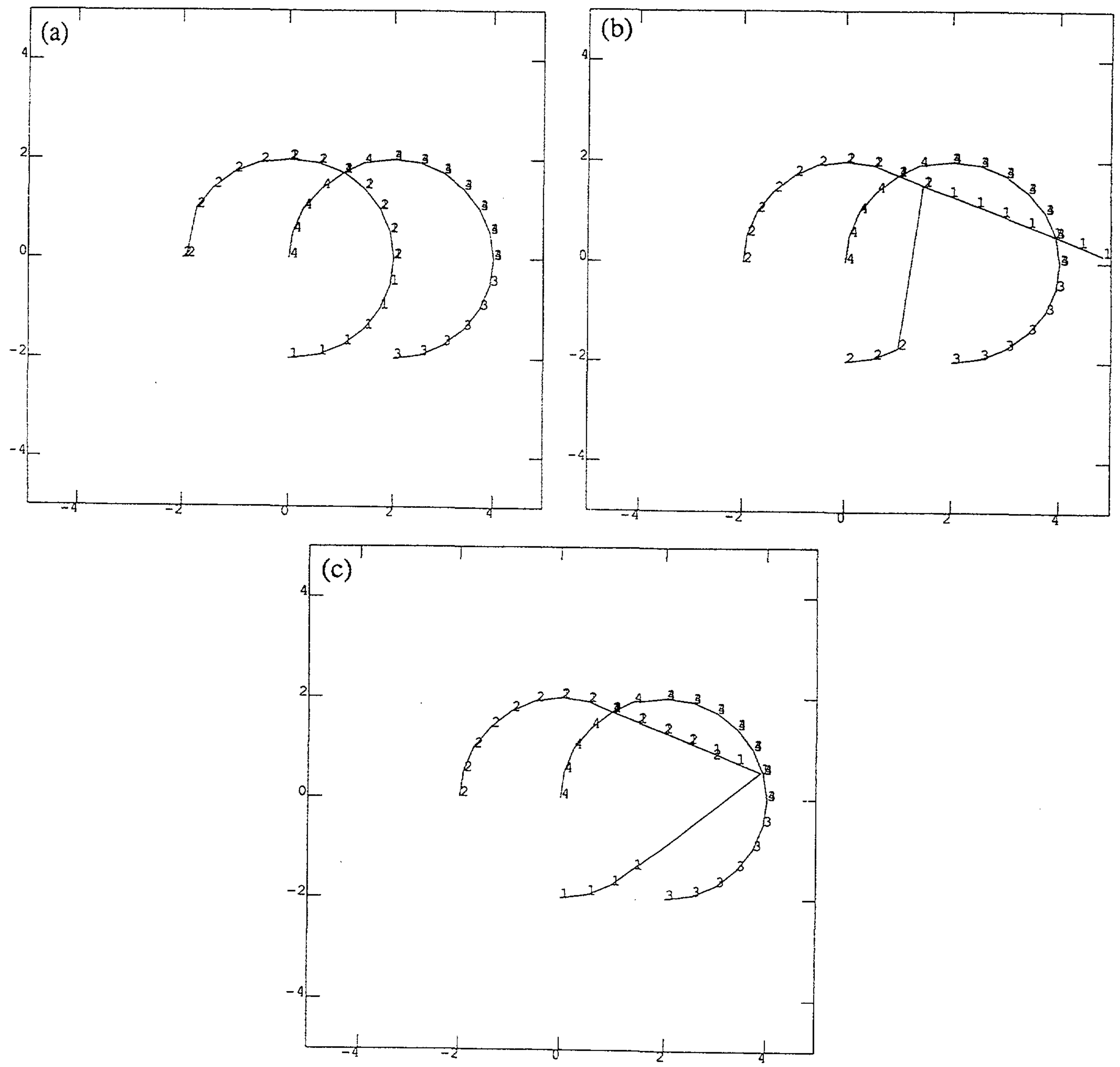


FIG. 18. Rotating disks. (a) Ideal trajectories. (b) Trajectories with *event* occlusion. (c) Trajectories without *event* occlusion. In these trajectories, the locations of two different points coincide in the image plane at different time instances, which is shown by overwriting one number on the other.

ACKNOWLEDGMENT

The authors are thankful to Dr. Donna Williams for her thoughtful comments on the earlier-draft of this paper.

REFERENCES

1. K. Gould and M. Shah, The trajectory primal sketch, in *Conference on Computer Vision and Pattern Recognition, June, 1989,* pp. 79–85.
2. D. H. Ballard and C. M. Brown, *Computer Vision,* Prentice–Hall, Englewood Cliffs, NJ, 1982.
3. I. K. Sethi and R. Jain, Finding trajectories of feature points in a monocular image sequence, *IEEE Trans. Pattern Anal. Mach. Intelligence,* **PAMI-9,** 1987, 56–73.
4. S. Ullman and A. Yuille, *Rigidity and Smoothness of Motion,* Technical Report A. I. Memo 989, MIT AI Lab, Cambridge, MA, 1987.
5. James J. Little, Henirich H. Bulthoff, and Tomaso Poggio, Parallel optical flow using local voting, in *Proceedings of Second ICCV, 1988,* pp. 454–459.
6. S. Ullman, *Interpretation of Visual Motion,* MIT Press, Cambridge, MA, 1979.
7. Michael Jenkin, Tracking three dimensional moving light displays, in *Proceedings, Workshop Motion: Representation Contr. Toronto, 1983,* pp. 66–70.
8. J. K. Aggarwal, L. S. Davis, and W. N. Martin, Correspondence procceses in dynamic scene analysis, in *Proc. IEEE* **69,** 1981, 562–571.
9. S. T. Barnard and W. B. Thompson, *Disparity Analysis of Images,* Technical Report 79-1, Computer Science Dept., University of Minnesota, 1979.
10. Lance R. Williams and Allen R. Hanson, Translating optical flow into token matches and depth from looming, in *International Conference on Computer Vision, December, 1988,* pp. 441–448.
11. P. Anandan, *Measuring Visual Motion from Image Sequences,* Ph.D. thesis, University of Massachusetts at Amherst, 1987.
12. Juyang Weng, Naredra Ahuja, and Thomas S. Huang, Two-view matching, in *International Conference on Computer Vision,* December, 1988, pp. 64–73.
13. K. Rangarajan, M. Shah, and D. Van Brackle, Optimal corner detector, *Comput. Vision Graphics Image Process.* **48,** 1989, 230–245.
14. I. K. Sethi and R. Jain, *Establishing Correspondence of Non-rigid Objects Using Smoothness of Motion,* Technical Report CRL-TR-10-84, University of Michigan Center for Research on Integrated Manufacturing, 1984.
15. J. E. Cutting, A program to generate synthetic walkers as dynamic point-light displays, *Behav. Res. Methods Instrum.* **10**(1), 1977, 91–94.