MOTION TRAJECTORIES *

Mubarak Shah, Krishnan Rangarajan and Ping-Sing Tsai
Computer Science Department
University of Central Florida
Orlando, FL 32816

September 14, 1992

Abstract

In this paper, we present a simple algorithm for selecting and linking *interesting* flow vectors across a sequence of frames for computing motion trajectories. We track tokens which have both interesting pixel gray values in the spatial domain *and* in the optical flow field in the temporal domain. This *and* operation effectively remove some redundant trajectories. Due to errors introduced during the computation of optical flow, and the linking of such flow vectors across a sequence of frames, the resultant trajectories are not always smooth. We discuss a Kalman filtering based approach for smoothing the trajectories.

Isolating the trajectories into sets belonging to individual objects is an important first step that should be taken before any type of shape or motion interpretation can be done. Therefore, we also discuss a simple algorithm for segmenting motion trajectories. When motion trajectories belonging to a single translating object are extended, they intersect at a single point called the Focus Of Expansion (FOE). If the motions of objects are assumed to be independent, each FOE represents one object. Therefore, FOE can be used to segment trajectories belonging to individual objects. We present a simple but highly robust algorithm for partitioning motion trajectories, which does not require the exact location of FOE, but uses some useful properties of FOE.

We have applied our method for computing and segmenting motion trajectories to a number of real sequences, and have obtained very encouraging results.

*The research reported here was supported by the Florida HiTech Council under grant 65 02 731, and the National Science Foundation under grant number IRI 8900798.

1 Introduction

Recently, there has been a growing interest in the use of extended image sequences for recovering motion and structure. Previous experience with approaches using two or three frames has revealed that these methods become very complex when extended to multiple frames. Multiple frames are often used to introduce multiple constraints on the solution, resulting in an over-constrained system. The problem of occlusion can also be dealt with naturally, because in most cases the object, once occluded, will reappear in next few frames. Interesting theoretical work related to the number of points required for a solution, the uniqueness of such a solution, and the effect of noise on the solution has been studied. While it is important to work on these theoretical problems, it is also important to use theory to recover structure and motion information from real scenes, and to be able to apply such 3D information in practical visual tasks. In a recent survey of related work on multi-frame feature based motion analysis, Price [9] has noted that only one fourth of the papers actually report results on sequences of real scenes. The common assumption in all these approaches is that correspondence between points in a sequence of frames–2D motion trajectories—is known, which has proven to be a difficult problem.

Computation of motion trajectories involves two main steps: detection of feature points in each frame, and correspondence of those feature points among frames. Corners, interest points, edges, straight lines, and intersections of straight lines have been used as tokens. Corners are suitable for man-made objects, polyhedra, and for scenes containing buildings. Edges and the intersection of lines are useful for domains with lines as the dominating structure. None of these token detectors perform well for general scenes without the tuning of various parameters. Either too many or too few features are detected.[1] The computation of motion trajectories is further compounded due to the combinatorial nature of the correspondence problem; that is, one has to select a single set of trajectories among the many possible [12, 10].

An alternate method for measuring 2D motion is optical flow. Unlike motion correspondence, optical flow does not require any tokens, and operates directly on gray level images. Significant progress has been made in optical flow methods, and a number of approaches exist which do a reasonably good job in computing optical flow for sequences of real scenes with natural motion (e.g. see [2, 6]). Recently, Barron, Fleet and Beauchemin [4], have implemented nine existing optical flow techniques and have presented a very good performance evaluation. Even though methods for recovering motion and structure from optical flow using second and third order derivatives have been reported, our aim in this paper is to use optical flow for computing motion trajectories, and subsequently use motion trajectories for recovering motion and structure. Since optical flow is dense in nature, we present a method for selecting some *interesting* flow vectors using the interest operator in both the spatial and temporal domains. These vectors are obtained in multiple frames and are linked to compute motion trajectories. Due to errors introduced during the computation of optical flow, and the linking of such flow vectors across a sequence of frames, the resultant trajectories are not always smooth. We discuss a Kalman filtering based approach for smoothing the trajectories.

It is rare to find scenes with a single moving object. In real scenes, one observes multiple moving objects with a variety of motions in assorted directions. In order to utilize the structure

[1] We believe that a single constraint that tokens should be *interesting* in the pixel gray levels is not enough. Quite a few of these tokens, which are irrelevant, can be filtered out by imposing a second constraint; a token should be interesting in the optical flow values as well. That is the approach followed in this paper.

from motion methods developed for a single isolated object in a realistic situation, one needs to segment the scene. Segmentation can be attempted in each frame using spatial information, but this has proven to be a non-trivial problem. Segmentation can also be performed using optical flow. For example, Adiv [1] considers the surface of an object to be composed of many piecewise planar patches, and presents a method for segmenting the optical flow field into connected sets of flow vectors where, each set is consistent with a rigid motion of a roughly planar patch. This method is based on the Ψ transform, which involves the generalized Hough transform technique for grouping flow vectors satisfying the same set of parameters. The disadvantage of this method, however, is that the resulting mosaic of many pieces must then be grouped or fused, on the basis of similar motion and structure, into larger surfaces using a multipass Hough transform technique. Therefore, it needs huge memory space, and computation time. Moreover, Adiv's method uses only two frames. We believe that segmentation using a large number of frames would be simpler and more reliable. In particular, we are interested in segmentation using extended motion trajectories, instead of optical flow. In fact, when motion trajectories belonging to a single translating object are extended, they intersect at a single point, called the Focus Of Expansion (FOE). If the motions of objects are assumed to be independent, each FOE represents one object. Therefore, FOE can be used to segment trajectories belonging to individual objects. We present a simple but highly robust algorithm for partitioning motion trajectories, which does not require the exact location of FOE, but uses some useful properties of FOE.

In short, given a sequence of real scenes, our aim is to use methods developed for recovering motion and structure of objects automatically, using possibly fewer parameters. We assume (and use Anandan's [2] algorithm) that there exists a good optical flow algorithm which gives reasonable results for real scenes. In this paper, we present a simple algorithm for selecting and linking *interesting* flow vectors across a sequence of frames for computing motion trajectories. Further, we outline an algorithm for segmenting such motion trajectories into groups belonging to individual moving objects. Finally, we assume that those motion trajectories can be used in methods for recovering motion and structure. Our emphasis in this work is on the modularity of the system. We envision three main modules for recovering motion and structure: the optical flow module, the trajectory generation and segmentation module, and the structure and motion computation module. We believe that optical flow, and structure and motion computation modules are fairly well established; in this paper we provide a missing link: a trajectory generation and segmentation module.

2 Computing Motion Trajectories

Optical flow is a displacement vector field in the image plane induced by the motion of objects, the observer, or both. In principle, the flow vectors in successive frames can be linked across a sequence of frames into motion trajectories. Since the optical flow is dense in nature, there will be too many such trajectories. Therefore, a few selected points obtained by Moravec's [8] interest operator, or any corner detector, can be tracked. Such tokens might represent some portions of the image which remain stationary throughout the sequence. Some of these tokens can be filtered out by introducing an *optical flow* constraint–tokens should be interesting in the optical flow field as well.

For a token to be interesting, its optical flow should differ from the mean optical flow around a small neighborhood by some amount proportional to its standard deviation. Since optical flow

is a vector quantity, we will compute the mean and standard deviation for its speed and direction separately in a small neighborhood. This is a very general criteria, which does not depend on many arbitrary thresholds. This criteria is also very close to our intuitive notion of similarity, that is, a token is similar to its neighbors if its value is close to the mean of their distributions. This criteria for identifying interest points in the optical flow field may appear to be sensitive to the outliers in the optical flow. Since the interest point is defined to be different from its neighbors, the outlier is always different from its neighbors. Therefore, we have two conflicting conditions to be satisfied: the point is not an outlier, and is interesting. One can define the point to be interesting if its optical flow is similar to its neighbors. However, that criteria will result in large numbers of redundant points, which we are attempting to reduce to start with. From our experimental results we have noticed that our criteria performs quite well. There was only one case, where the outlier was selected as an interest point, that point was subsequently removed during the segmentation step of our algorithm.

One can only detect the interest points in the first frame, filter some points by applying the optical flow constraint, and link the flow vectors for the remaining frames. Due to the extra smoothing step in Anandan's method, the optical flow is not always accurate; hence, linking the flow vectors without verifying whether the tracked point in successive frames indeed is an interest point may amplify the error. Therefore, while tracking a point from the current frame to the next frame, the small neighborhood around the predicted location of the interest point will be searched, and the flow vector will be linked to the interest point closest to the central point. In case no interest point is observed in the neighborhood, the predicted location is used. Figure 1 demonstrates this correction step. The points labeled A are the correct locations obtained by manually tracking the point on the Soccer ball. The trajectory is shown by the continuous curve. The trajectory B, shown with a broken curve and +'s, is obtained by linking flow vectors with the correction step. Finally, trajectory C, shown by a broken curve and unfilled squares, is obtained without the correction step. The total point to pint Euclidean distance between A and B is 26, and the total Euclidean distance between A and C is 141. Therefore the corrected trajectory is much closer to the true trajectory, even though visually the trajectory C appears closer to the trajectory A. In all the experiments performed so far, we have rarely encountered a case with more than two tokens in a small neighborhood. In most cases a unique token close to the central point always exists. This is partly due to the optical flow constraint, which is able to remove the tokens with similar flow vectors.

Another obvious possibility for introducing the optical flow constraint, which does not work, is to employ Moravec's interest operator to the optical flow field. Moravec's interest operator involves two steps: First, each pixel is assigned a minimum of its variance in gray levels computed in four directions (horizontal, vertical, diagonal, and ant-diagonal). Second, the point is declared interesting if its minimum variance is local maximum in a 12×12 window considering overlapping neighborhoods. Due to this large window the tokens are delocalized in some cases. This delocalization is even more serious when the gray level tokens are *anded* together with the optical flow tokens.

Our method for linking optical flow vectors into motion trajectories is similar in spirit to the method proposed by Williams and Hanson [14] for translating optical flow into token matches; however, it is different in many aspects. Williams and Hanson solve line correspondence in a pair of frames using optical flow information. First, they identify line segments in each frame using Boldt's [5] line extraction algorithm. Then they compute optical flow using Anandan's algorithm.

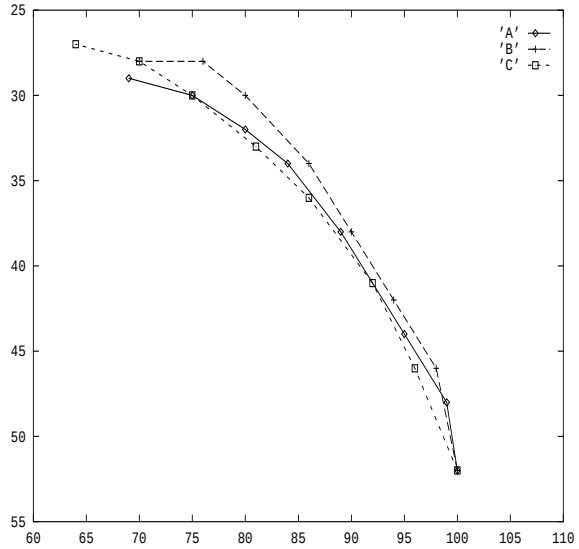

Figure 1: The points labeled A are the correct locations obtained by manually tracking the point on the Soccer ball. The trajectory is shown by the continuous curve. Trajectory B, shown with broken curve and +'s, is obtained by linking flow vectors with the correction step. Trajectory C, shown by a broken curve and unfilled squares, is obtained without the correction step.

Next, for each point corresponding to a line in first frame, they fit a straight line (using least squares fit) to the end points of the flow vectors. That straight line is considered a predicted straight line. A rectangular area around the predicted straight line is used as a search area for the correspondence. They pose the correspondence problem as the minimum cover of a bi-partite graph. The nodes in such a graph are the straight lines in both frames. The links between nodes connect line segments from the first frame's token set, to all candidate matches retrieved from the second frame. They also mention an extension of this method for multiple frames by running it repeatedly on successive frames, creating a directed acyclic graph and representing the splitting and merging of line segments over time. In another related paper by Sawhaney *et al* [11] the intersections of straight lines (corner points) are tracked, and conic sections are fitted to the trajectories for computing the rotation of the axis.

Williams and Hanson's method is very interesting, but highly complex. Although they rely on straight lines as tokens, in some scenes it is rare to find straight lines, and the method for fitting straight lines itself is involved. An additional step for fitting straight lines to the end points of optical flow is therefore needed. Since the straight line as a token is highly unstable in its attributes across frames, the merging and splitting of line segments should be considered. Our method is very simple, and gives quite good results. We track flow vectors which are interesting in the spatial as well as the temporal domain (thus avoiding too many redundant trajectories), and link them into motion trajectories.

Kenner and Pong [7] describe two related methods for computing long image sequence flow. In the first method they detect tokens in each frame of a sequence, and store tokens obtained from all frames in a composite frame. They assume the motion to be small, consider adjacent tokens to be in correspondence, and fit straight lines to the adjacent tokens using the Hough transform. This step implies that the authors are only interested in translation trajectories. In the second

method, Kenner and Pong compute optical flow for the first two frames, and use the flow vectors to constrain the search to a small region. If no match is found for a particular token, the search for that token is discontinued. Our method is more general, and will work with the output from any optical flow method. Since we track the tokens which are interesting in the spatial and temporal domain, our search space is very small. Moreover, our method is not limited to only translation motion. It will work for any type of motion, including rotation.

3 Smoothing Trajectories using Kalman Filter

The motion of most objects in nature is continuous, and the resultant motion trajectories are piecewise smooth. However, we have noticed that the trajectories obtained by the method discussed in the previous section may not be smooth. This is due to two reasons: First, the optical flow is not always accurate, and second some error is introduced during the linking of optical flow vectors. These two sources of error can integrate over time and may result in trajectories being not smooth. Therefore, we introduce the Extended Kalman Filter [3] (EKF) as a post-processor for smoothing the trajectories. Kalman filter has been extensively used in solving several vision problems. For instance see [13] for a recent work.

We will model the motion of trajectories as a 2-D ballistic motion. The position (x^t, y^t) of a trajectory at time t is given as

$$x^t = \frac{1}{2}a_x t^2 + v_x t + x_0 \tag{1}$$

$$y^t = \frac{1}{2}a_y t^2 + v_y t + y_0 \tag{2}$$

where a_x and a_y are accelerations in the x and y directions, v_x and v_y are velocities in the x and y directions, and (x_0, y_0) is the initial position. We will assume that the initial position is known (We will use the first point of the trajectory as the initial position). Therefore, we will be dealing with 4 unknown parameters (a_x, a_y, v_x, v_y), and the measurement vector (x^t, y^t). We can consider the function $f(b, c) = (0, 0)$, where $b \in R^2$ is the position of the trajectory, and $c \in R^4$ is the unknown parameter vector:

$$f(b,c) = (x^t - \frac{1}{2}a_x t^2 - v_x t - x_0,\ y^t - \frac{1}{2}a_y t^2 - v_y t - y_0) = (0,0) \tag{3}$$

The problem is to find the best estimate $\hat{c}$ of c given the function f and the measurement vector $\hat{b}^t$. This is very suitable for the Extended Kalman filtering approach. The recursive equations of the Kalman filter provide new estimates $\hat{c}^t$ of $\hat{c}^{t-1}$, and S^t of S^{t-1} as follows:

$$\begin{aligned} \hat{c}^t &= \hat{c}^{t-1} + K^t(Y^t - M^t\hat{c}^{t-1}) \\ K^t &= S^{t-1}(M^t)^T(W^t + M^t S^{t-1}(M^t)^T)^{-1} \\ S^t &= (I - K^t M^t)S^{t-1} \end{aligned}$$

where

$$Y^t = -f^t(\hat{b}^t, \hat{c}^{t-1}) + \frac{\partial \hat{f}^t}{\partial c}c^{t-1},$$

$$M^t = \frac{\partial \hat{f}^t}{\partial c},$$
$$W^t = \frac{\partial \hat{f}^t}{\partial b} \Lambda^t \frac{\partial \hat{f}^t}{\partial b}^T,$$

Λ^t is the covariance matrix for the measurement vector b^t which is assumed to be known, and T denotes the transpose of the matrix. The proportional factor K^t is known as the Kalman gain, and S^t is the covariance matrix associated with the error in the estimate. At the end of each iteration, the trajectory can be regenerated based on the parameter vector c.

4 Results

The proposed algorithm for computing motion trajectories was tested on several sequences of images. In the first example (shown in Figure 2), we videotaped the motion of three toys: a Jeep, a Car and a Truck. The Jeep moves from left to right, the Car moves diagonally from the bottom right corner to the top left corner, and the truck moves from left to right in a direction slightly towards the upper right corner. Five frames from the sequence were digitized, and optical flow was computed for the first two frames as shown in Figure 2(f). The Moravec operator selected 14 points, ten points corresponding to the moving objects, and four stationary points corresponding to the date and time at the lower right corner of the image. The optical flow constraint was able to remove all stationary points, and one point corresponding to the top left point on the car. The optical flow distribution around that point is shown in Figure 3(b). Since the optical flow at the central point is similar to its neighbors, it was not selected. Note that the direction as well as the speed is considered in determining the similarity of flow field. This can be explained more clearly by referring to Figure 3(d), in which the polar coordinates for the flow vectors are used. The mean of the flow vectors is shown by 'X', and the box around the mean (determined by the standard deviations of the speed and direction, in this example, we use ± 1 standard deviation) identifies a rectangular region in the speed-direction space; the flow vectors in that region are considered to be similar. Since the central point shown by 'P' is inside the box, it is considered similar to its neighbors, and is not selected. The distribution of flow vectors around the right-most point on the truck which was selected is shown in Figure 3(a), and its corresponding polar plot is shown in Figure 3(c). It is clear from Figure 3(c) that the flow vector of this point lies outside the rectangular box, hence it is different from surrounding points. The trajectories computed by linking the optical flow vectors for the points shown in 2(d) and 2(e) respectively are shown in 2(g) and 2(h). The smoothed trajectories after applying the Kalman filter are shown in 2(i). These trajectories appear smoother than the trajectories shown in parts (g) and (h).

The results for the Soccer ball sequence are shown in Figure 4. The ball is rotating counter-clockwise around the Z- axis. The Moravec interest operator picked up 52 points, 12 of which were filtered out by the optical flow constraints. The optical flow corresponding to six of those points which were close to the center was too small (indicated by Figure 4(f)), and the remaining six points had optical flow very similar to their neighbors.

For the Cones sequence (shown in Figure 5), the camera was moved inside a pathway, and the cones were stationary. The trajectory for the cone on the left (right) of the camera was moved to the left (right). The point corresponding to the left-most cone moved out of view after frame 5; our program stopped linking the trajectory from that frame. The point corresponding to the ceiling

light at the top moves out of the view in latter frames. Since that point is very close to the border of the image, the optical flow estimate was incorrect there, and the program continued to link the flow vectors in the wrong direction.

5 Segmentation

Isolating the trajectories into sets belonging to the same object is an important first step that should be taken before any type of shape or motion interpretation can be done. The trajectories can be broadly classified into three groups, depending on the type of motion: translation trajectories, rotation trajectories, and translation and rotation trajectories. A typical scene may contain multiple objects; each object gives rise to a set of trajectories which could be one of the above-mentioned types. In general, all three types of motion could be present in a scene. The ideal segmentation algorithm will be the one which is able to segment the trajectories in a scene like this. In this section, we present a method for segmenting trajectories in a scene, where the motion of objects in Z direction is small and there are no rotating objects. All points lying on an object that undergoes only translation have the same velocity in the X, Y and Z directions. By the definition of the FOE (Focus of Expansion), the trajectories of points belonging to the same object will have the same FOE. Associated with each pair of trajectories is a FOE. If all pairs of trajectories considered from three trajectories in the scene have the same FOE, by the principle of *non-accidentalness*, we can say that the three trajectories belong to the same object. Under this assumption, if trajectories p, q and r have the FOE $\vec{\vartheta}_1$, and if trajectories p, s and t have the FOE $\vec{\vartheta}_2$, then $\vec{\vartheta}_1$ and $\vec{\vartheta}_2$ should be the same, and trajectories p, q, r, s and t belong to the same object. We use this fact in our segmentation algorithm. Hence, one of the basic assumptions of this algorithm is that there are at least three trajectories for each of the objects present in the scene. It is possible that more than one object may have the same velocity at a particular time instant. In that case, all trajectories belonging to these objects are likely to be grouped as belonging to the same object. To tackle this problem, our algorithm looks at the whole span of the trajectories at all available time instants. Our algorithm will classify trajectories p, q and r as belonging to different objects, even if they have the same pairwise FOE at all time instants except the one.

This simple segmentation algorithm was tried over a synthetic sequence having two objects, a Cube, and a Pyramid, as shown in figure 6. The objects were moved such that they had uniform acceleration. The Cube had an initial velocity of $v_X = 3$, $v_y = 2$, $v_Z = 1$, and the Pyramid had $v_X = 3.1$, $v_Y = 2.1$, $v_Z = 1.1$. The corner points of these two objects were tracked over a sequence of frames. Even though at time t_2 they have the same velocity, since our segmentation algorithm looks at the entire span of the trajectories, it was able to segment the trajectories correctly. The results are shown in Figure 6(c) and Figure 6(d).

The above segmentation algorithm makes use of the fact that all the trajectories belonging to the same object will have the same FOE. However, in real images this might not be true, because there may be some errors introduced due to the optical flow and feature detection. Therefore, we will discuss a second algorithm which does not require exact computation of FOE, but uses some useful properties of FOE.

Let us consider the trajectories of three arbitrary feature points located on an object. Under ideal conditions, these three trajectories will pass through a common FOE when extended. Due to small errors in the trajectories generation, these three trajectories when extended will form a triangle, and we shall call this the *FOE triangle*. There is an FOE triangle associated with every set

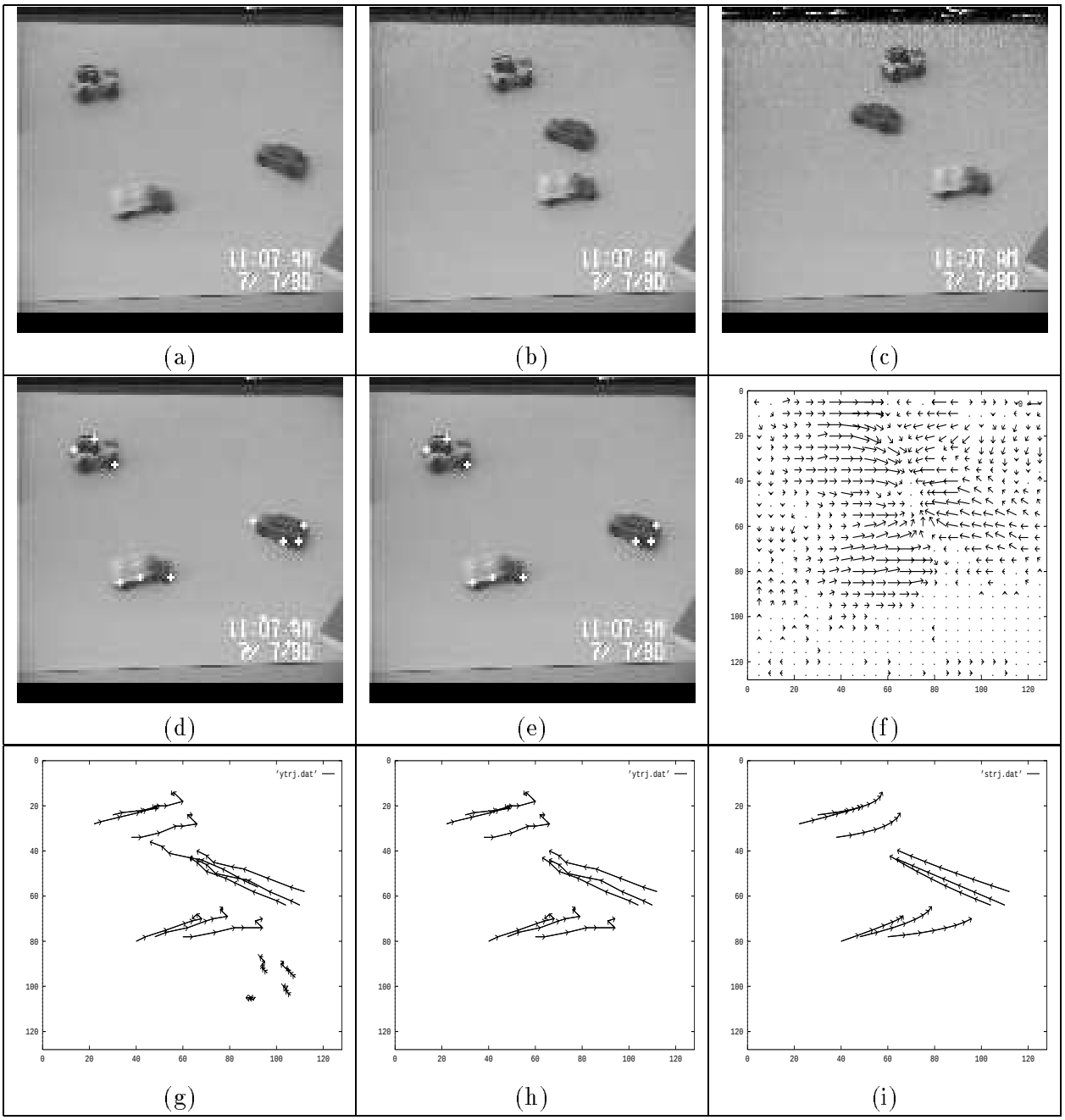

Figure 2: Results for the Car Sequence. (a) Frame 1. (b) Frame 5. (c) Frame 8. (d) Moravec interest points (total 14 points) superimposed on Frame 1. (e) After including optical flow information, only 9 points were left. (f) Optical flow for the first two frames. (g) Trajectories before including optical flow constraint. (h) Trajectories after including optical flow constraint. (i) Trajectories after smoothing.

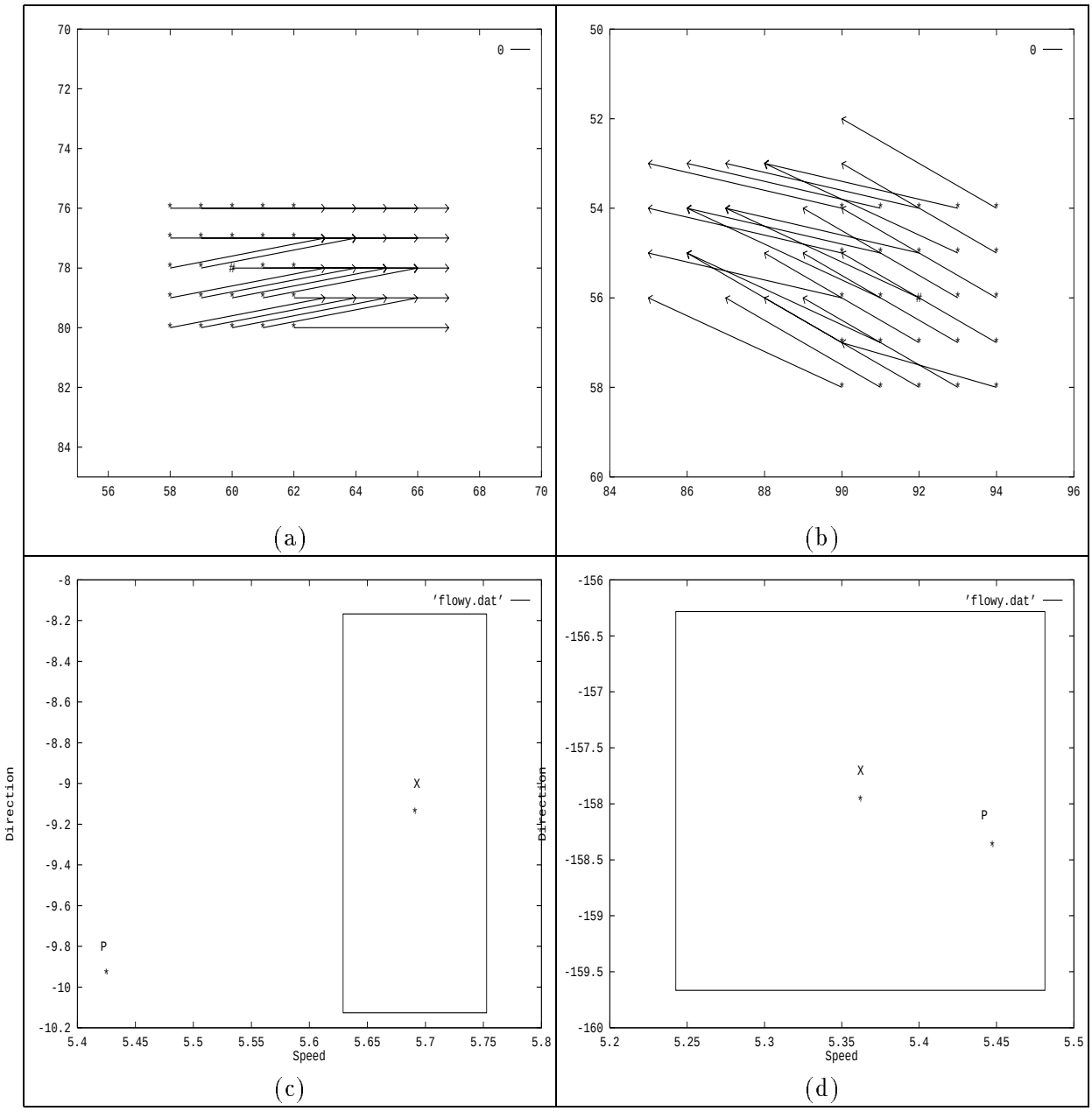

Figure 3: Distribution of flow vectors for the Car Sequence. (a) The flow vectors around the right most-point on the truck $(60, 78)$, which was selected by the optical flow constraint. (b) The flow vectors around point $(92, 56)$, which was not selected by the optical flow constraint. (c) The distribution of optical flow around point $(60, 78)$ in the speed–direction space. The mean optical flow is shown by 'X'. Since the optical flow of central point denoted by 'P' lies outside the rectangular region determined by the standard deviations of speed and direction, it is different from its neighbors. Therefore, it was selected by the optical flow constraint. (d) The distribution of optical flow around point $(92, 56)$ in the speed–direction space. The mean optical flow is shown by 'X'. Since the optical flow of central point denoted by P lies inside the rectangular region determined by the standard deviations of speed and direction, it is similar to its neighbors. Therefore, it was not selected by the optical flow constraint.

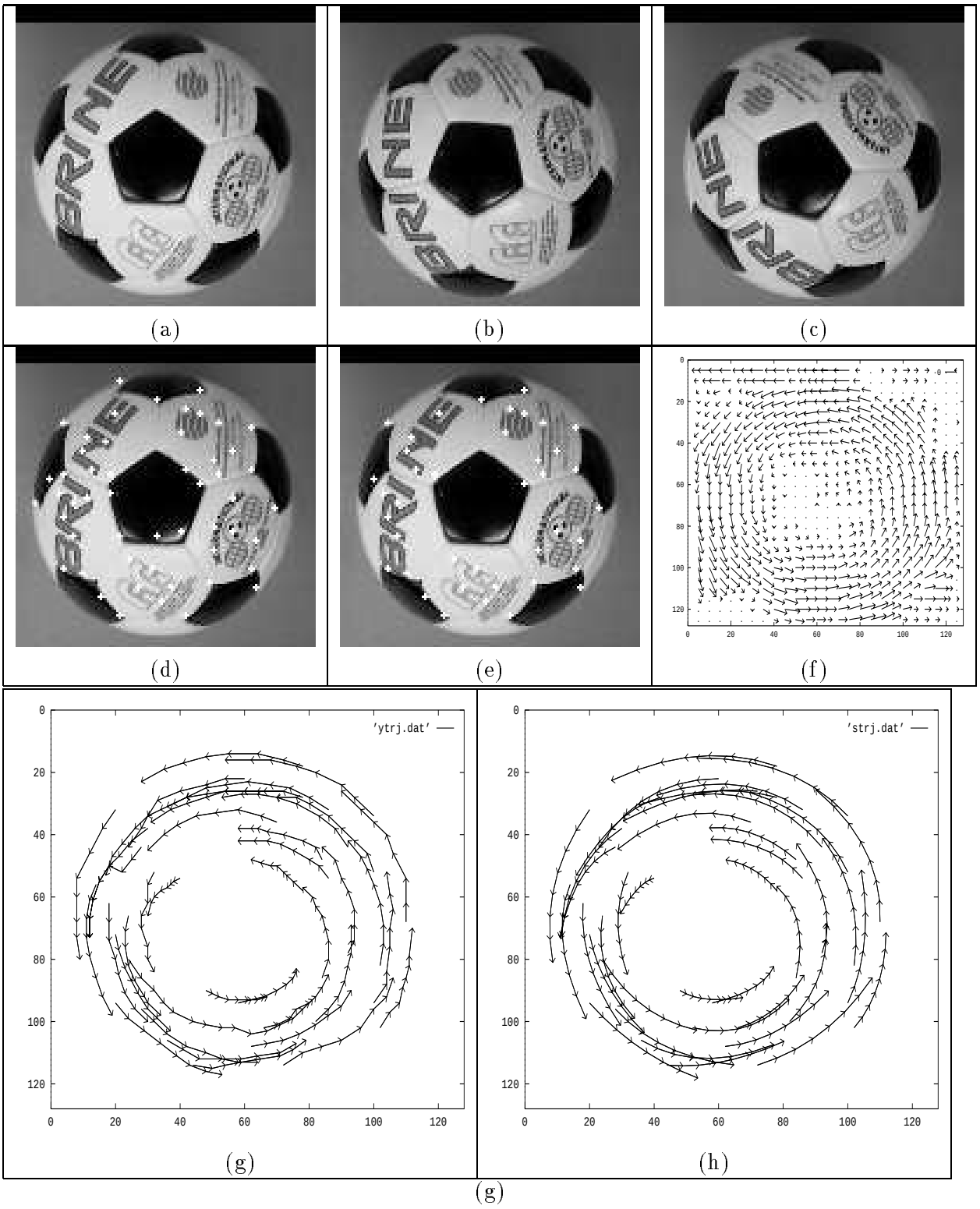

Figure 4: Results for the Soccer Ball Sequence. (a) Frame 1. (b) Frame 5. (c) Frame 8. (d) Moravec interest points (total 52 points) superimposed on Frame 1. (e) After including optical flow information, 40 points were left. (f) Optical flow for the first two frames. (g) The resultant trajectories . (h) The resultant trajectories after smoothing .

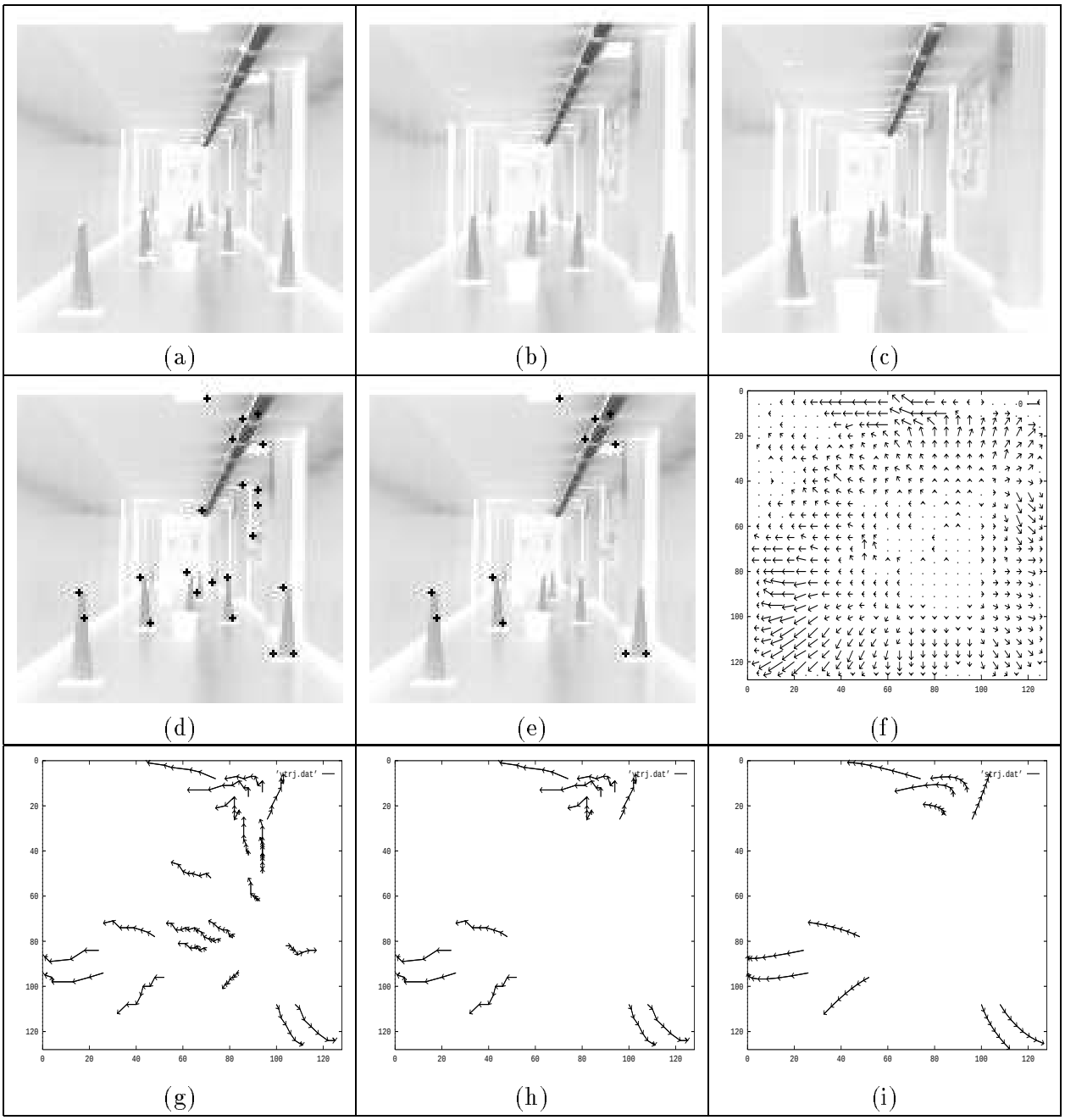

Figure 5: Results for the Cones Sequence. (a) Frame 1. (b) Frame 5. (c) Frame 8. (d) Moravec interest points (total 22 points) superimposed on Frame 1. (e) After including optical flow information, 10 points were left. (f) Optical flow for the first two frames. (g) Trajectories before including optical flow constraint. (h) Trajectories after including optical flow constraint. (i) Trajectories after smoothing.

of three trajectories. The FOE triangle has some nice properties which can be used in trajectory segmentation, assuming that the errors introduced in the trajectory generation process are small, and the distance of the FOE from the position of the feature points in the image is large.

Let p_1^1, p_2^1, p_3^1 be three points in frame 1, as shown in Figure 7. Let p_1^2, p_2^2, p_3^2 be their corresponding points in frame 2. Let f be the point of intersection of these three trajectories. Due to errors in optical flow and feature detection, let points p_1^2, p_2^2, p_3^2 be detected at q_1^2, q_2^2, q_3^2 respectively. Let angles $q_1^2 p_1^1 p_1^2$, $q_2^2 p_2^1 p_2^2$, $q_3^2 p_3^1 p_3^2$ be $d1$, $d2$ and $d3$ respectively. Let $f_{12} f_{13} f_{23}$ be the FOE triangle formed. We will show that two of the angles of this triangle will be small. In general if the Z component of velocity is small, the angles r, s formed by the ideal trajectories at the FOE f will be small.

$$\begin{aligned} f_{13} f_{23} f_{12} + d2 &= s + d3 \\ f_{13} f_{12} f_{23} + d1 &= r + d2 \end{aligned}$$

Since r, s, $d1$, $d2$, and $d3$ are all small, the two angles $f_{13} f_{23} f_{12}$ and $f_{13} f_{12} f_{23}$ will also be small. We will call this an *angle constraint*. It can be shown that using this constraint, the probability of accidentally grouping any three arbitrary trajectories is small, if a small threshold for angles is used. Assume that all angles are equally likely. Since the sum of angles $f_{13} f_{23} f_{12}$ and $f_{13} f_{12} f_{23}$ can vary from 0 to 180, the probability of accidentally grouping the trajectories is $x/180$, where x is the angle threshold. This probability for a small threshold, say $x = 20$, becomes $20/180 = 0.111$. In case of set of trajectories greater than 3, we check for each 3-combination of this group. Hence the probability of grouping 4 or more trajectories is even smaller.

We will be making two additional assumptions in our segmentation algorithm. First, we will assume that the distance between feature points lying on the same object is small; this will be termed the *distance constraint*. Second, all trajectories belonging to the same object have the same sign of motion along the x and y axes; this will be called the *sign constraint*. The second assumption is a necessary condition. For example, for the trajectories shown in figure 7 to belong to the same object, each term in the vector difference of $q_1^2 - p_1^1$, $q_2^2 - p_2^1$ and $q_3^2 - p_3^1$ should have the same sign along the x and y axes.

We tried the above segmentation algorithm over a sequence of a real scene shown in Figure 8. This scene contains four moving objects: a Cardboard Box, a Rubic Cube, a Metal Box, and a Black Object. Points a, b and c on Cardboard Box, points d, e, f and g on Rubic Cube, and points h, i, j, k, l and m on Metal Box were tracked. The algorithm came up with the correct segmentation. By strict order analysis, the order of this algorithm is exponential. In this real scene, there are 13 trajectories. The total possible number of 3 combinations is $\binom{13}{3} = 286$, but only 25 combinations survived the first step. The number of 4-combinations obtained for this set was 16, the number of 5-combinations was 6, and there was only one 6-combination. Note that none of the invalid 3-combinations survived the first step. Figure 8(i) shows some sample numerical values of the terms computed in steps 1(b) and 1(c) of the segmentation algorithm using FOE triangle. The values for these combinations are much higher than the values of the trajectory groups which belong to the single object.

We also applied this algorithm to another sequence of real scenes shown in Figure 9, consisting of 5 frames, with 4 moving objects: a Rubic Cube, a Toy Truck, a Black Box and a Metal box. Points a, b and c on the Rubic Cube, points d, e, f and g on the Toy Truck, points h, i, j and k on

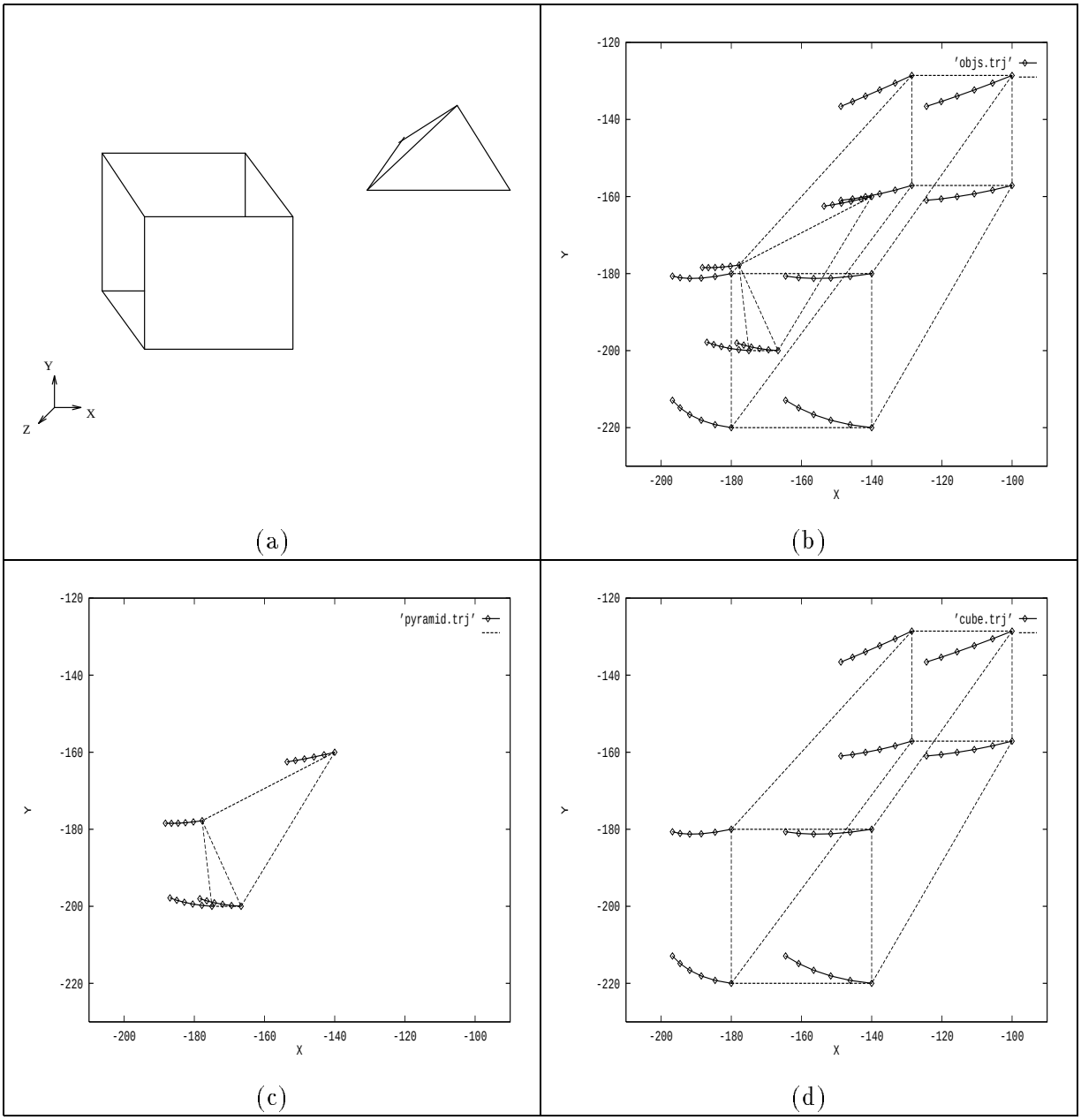

Figure 6: (a) A scene with two objects: a Cube and a Pyramid. The cube has a velocity of $v_X = 3$, $v_Y = 2$ and $v_Z = 1$. The Pyramid has a velocity of $v_X = 3.1$, $v_Y = 2.1$ and $v_Z = 1.1$. (b) The trajectories traced out by the objects with the objects superimposed on the location of the first frame. The corner points on the objects were picked and tracked. There are 8 corners on the Cube, and 4 corners on the Pyramid. (c)-(d) The segmentation of trajectories as belonging to different objects. Even though at some time instants the objects show the same velocity, the segmentation algorithm looks at the entire frame sequence and segments the trajectories correctly. (c) Trajectories of Pyramid. (d) Trajectories of Cube.

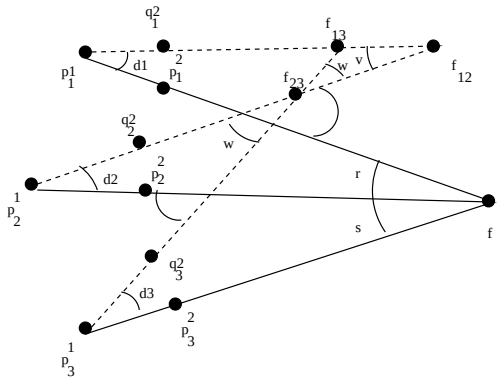

Figure 7: When there is no error, true FOE will be the point f. p_1^1, p_2^1, p_3^1 are three points in frame 1, while p_1^2, p_2^2, p_3^2 are their corresponding points in frame 2. Due to small errors in optical flow and feature point detection, p_1^2, p_2^2, p_3^2 are detected at q_1^2, q_2^2, q_3^2, which are in the neighborhood of p_1^2, p_2^2 and p_3^2, and FOE triangle $f_{13}f_{12}f_{23}$ is formed. Hence angles d_1, d_2 and d_3 are small. If time to collision is large, angles r and s will also be small.

SEGMENTATION ALGORITHM USING FOE TRIANGLE

1. /* Find all three combinations of compatible trajectories. */
   For every three combinations of trajectories do

   (a) Form the FOE triangle, by extending the trajectories and finding the intersection points. There will be $(n-1)$ FOE triangles, frames 1 through $n-1$.

   (b) Sum two smallest angles (show as w and v in figure 7) in each FOE triangle. Normalize the sum by dividing $\pi.(n-1)$. Since the sum of the angles in a triangle is π, there are $(n-1)$ such triangles.

   (c) The three points corresponding to trajectories under consideration form a triangle (show as q_1^2, q_2^2, and q_3^2 in figure 7) in the image plane for each frame. The perimeter of this triangle is equal to the sum of the pairwise distance between the points. There will be $n-1$ such triangles. Find the sum of the perimeters of these $n-1$ triangles. Normalize the sum by dividing it by $3 \times (n-1).\sqrt{2}.size\ of\ the\ image$. Since there are 3 sides in each triangle, there are total of $(n-1)$ such triangles, and the maximum possible length of a side of any triangle in an image is $\sqrt{2}.size\ of\ the\ image$.

   (d) If for all the frames, these three trajectories agree in the sign of motion in both the x and the y directions, and if the terms computed in (b) and (c) are each small, the trajectories in this three-combination are considered likely to be on the same object, otherwise not.

2. /* Group 4 or more points together which lie on the same object. */
   An arbitrary set of m points, $m \geq 4$, are compatible if the m possible combinations with $(m-1)$ points each are all compatible combinations. This is a recursive definition which terminates at $m = 3$, and the valid 3 combinations are supplied by the results from the previous step.

3. The output from the previous step is groups of trajectories of different sizes and costs (sum of terms in b, c in step 1). A subset of these groups is selected such that: the groups are disjoint, are of low cost, and are of large size.

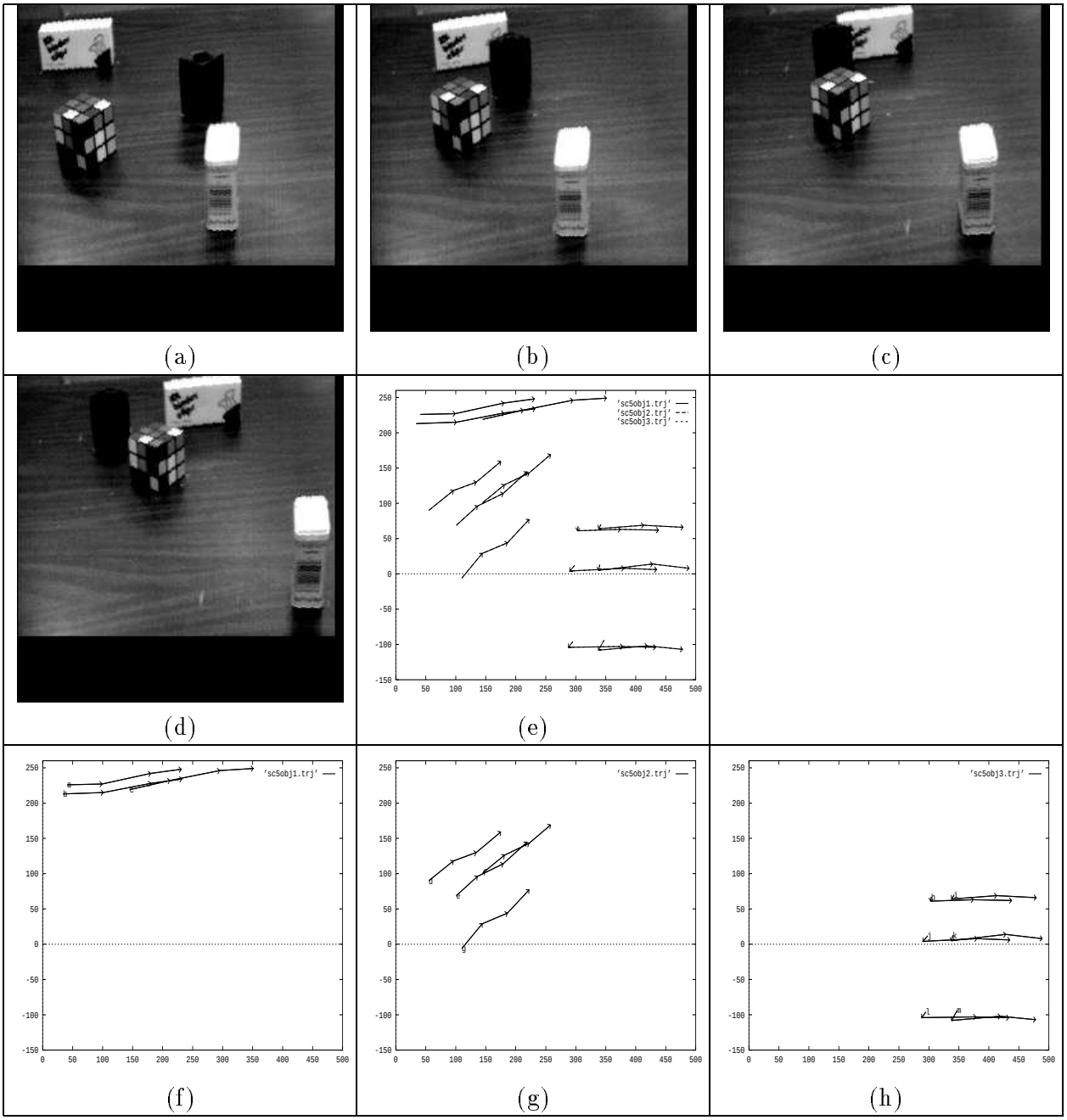

Figure 8: Results of segmentation algorithm. (a)-(d) A sequence of a real scene with four moving objects: A Cardboard Box, a Rubic Cube, a Metal Box and a Black Object. Points a, b and c on Cardboard Box, points d, e, f and g on Rubic Cube and points h, i, j, k, l and m on Metal Box were tracked. The proposed algorithm came up with the correct segmentation. (e) The trajectories of the marked points over 4 frames. (f) The group of trajectories belonging to Cardboard Box. (g) The group of trajectories belonging to Rubic Cube. (h) The group of trajectories belonging to Metal Box. (i) (Shown on next page.) Table showing some sample values for the terms computed in steps 1(a) and 1(b) of the algorithm. The first group shows the values of terms for trajectories a, b, and c, of the Cardboard Box . The next group consists of four 3-combinations of trajectories d, e, f, g of Rubic Cube. Similarly, the twenty 3-combinations for Metal Box are shown next. The last group shows the value of terms for trajectories which do not belong to any single object. The values for these combinations are much higher than the values of the trajectory groups which belong to the single object.

group	combinations	angle term	distance term
I	c b a	0.027364	0.281851
II	f e d	0.019996	0.242857
	g e d	0.044289	0.279342
	g f d	0.043931	0.348182
	g f e	0.038475	0.250298
III	j i h	0.073015	0.187118
	k i h	0.061171	0.200259
	k j h	0.082106	0.215616
	k j i	0.069718	0.215569
	l i h	0.067763	0.439357
	l j h	0.071511	0.422966
	l j i	0.029706	0.459424
	l k h	0.076854	0.444081
	l k i	0.068117	0.448836
	l k j	0.075177	0.319245
	m i h	0.057010	0.446013
	m j h	0.077945	0.438348
	m j i	0.030945	0.456505
	m k h	0.059404	0.448311
	m k i	0.045471	0.434767
	m k j	0.069718	0.330723
	m l h	0.072694	0.524705
	m l i	0.029344	0.534407
	m l j	0.036404	0.366869
	m l k	0.068117	0.371562
IV	e c a	0.156045	0.468729
	e c b	0.155134	0.449949
	f d a	0.138729	0.416527

(i)

Figure 8 (continued).

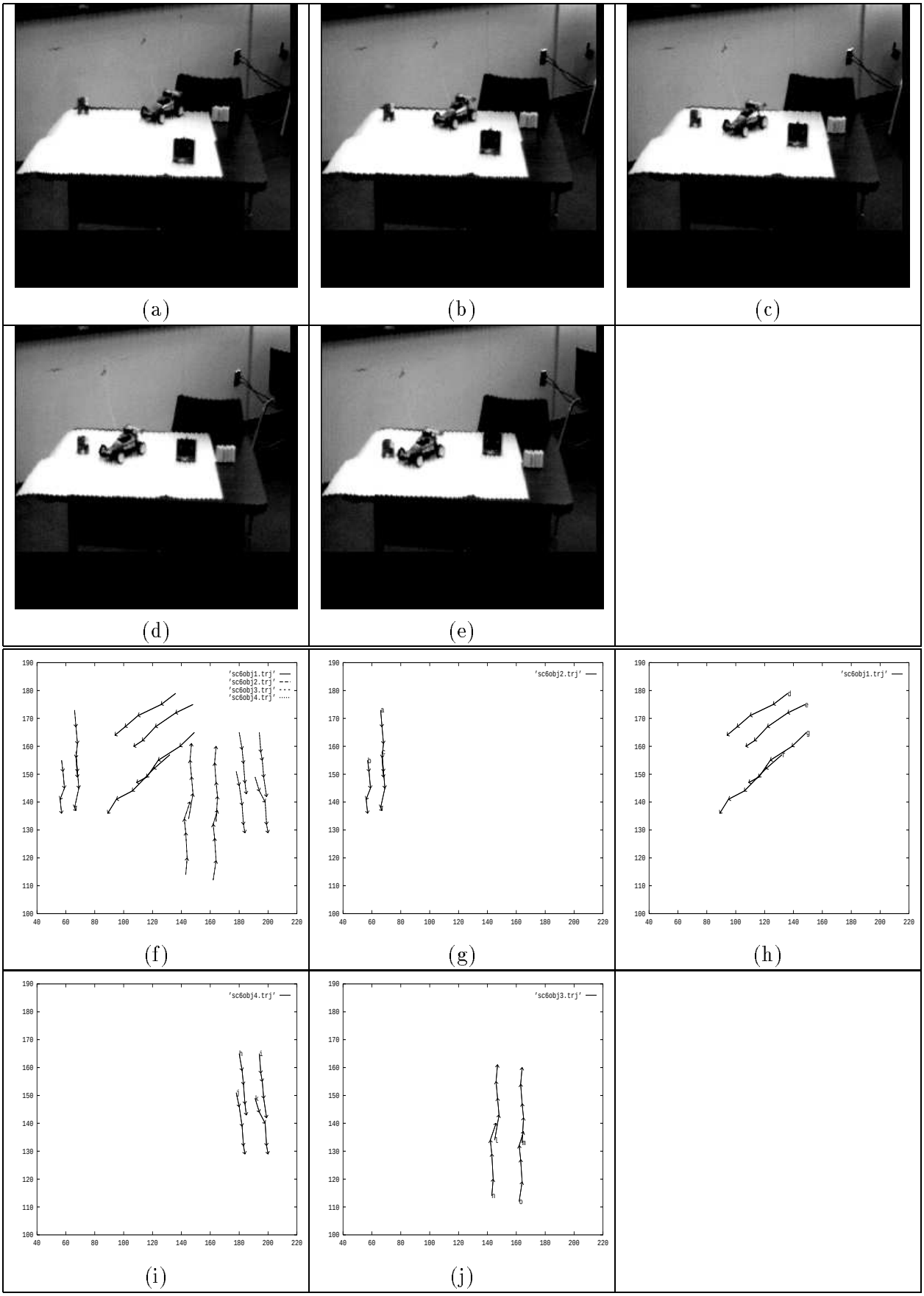

Figure 9: See caption on the next page.

group	combination	angle term	distance term
I	c b a	0.094869	0.056598
II	f e d	0.082760	0.074839
	g e d	0.053180	0.058921
	g f d	0.089231	0.079018
	g f e	0.093904	0.064600
III	j i h	0.061739	0.075492
	k i h	0.056865	0.083152
	k j h	0.052990	0.086570
	k j i	0.058283	0.084060
IV	n m l	0.036827	0.053364
	o m l	0.087821	0.060584
	o n l	0.071666	0.061325
	o n m	0.078736	0.060597
V	i h c	0.140803	0.306295
	l i c	0.156060	0.334388
	l j a	0.143287	0.337644
	m h b	0.157129	0.381925

(k)

Figure 9: Results of segmentation algorithm. (a)-(e) A sequence of a real scene with four moving objects: a Rubic Cube, a Toy Truck, a Black Box, and a Metal Box. Points a, b and c on Rubic Cube, points d, e, f and g on Toy Truck, points h, i, j and k on Black Box and points l, m, n and o on Metal Box were tracked. The proposed algorithm came up with the correct segmentation. (f) The trajectories of the marked points over 5 frames. (g) The group of trajectories belonging to Rubic Cube. (h) The group of trajectories belonging to Toy Truck. (i) The group of trajectories belonging to Black Box. (j) The group of trajectories belonging to Metal Box. (k) Table showing some sample values for the terms computed in steps 1(a) and 1(b) of the algorithm. The first group shows the values of terms for trajectories a, b, and c, of Rubic cube. The next group consists of four 3-combinations of trajectories d, e, f, g of the Toy Truck. Similarly, the four 3-combinations for Black Box, and Metal Box are shown next. The last group shows the value of terms for trajectories which do not belong to any single object. It is clear that the values for these combinations are much higher than the values of the trajectory groups which belong to the single object.

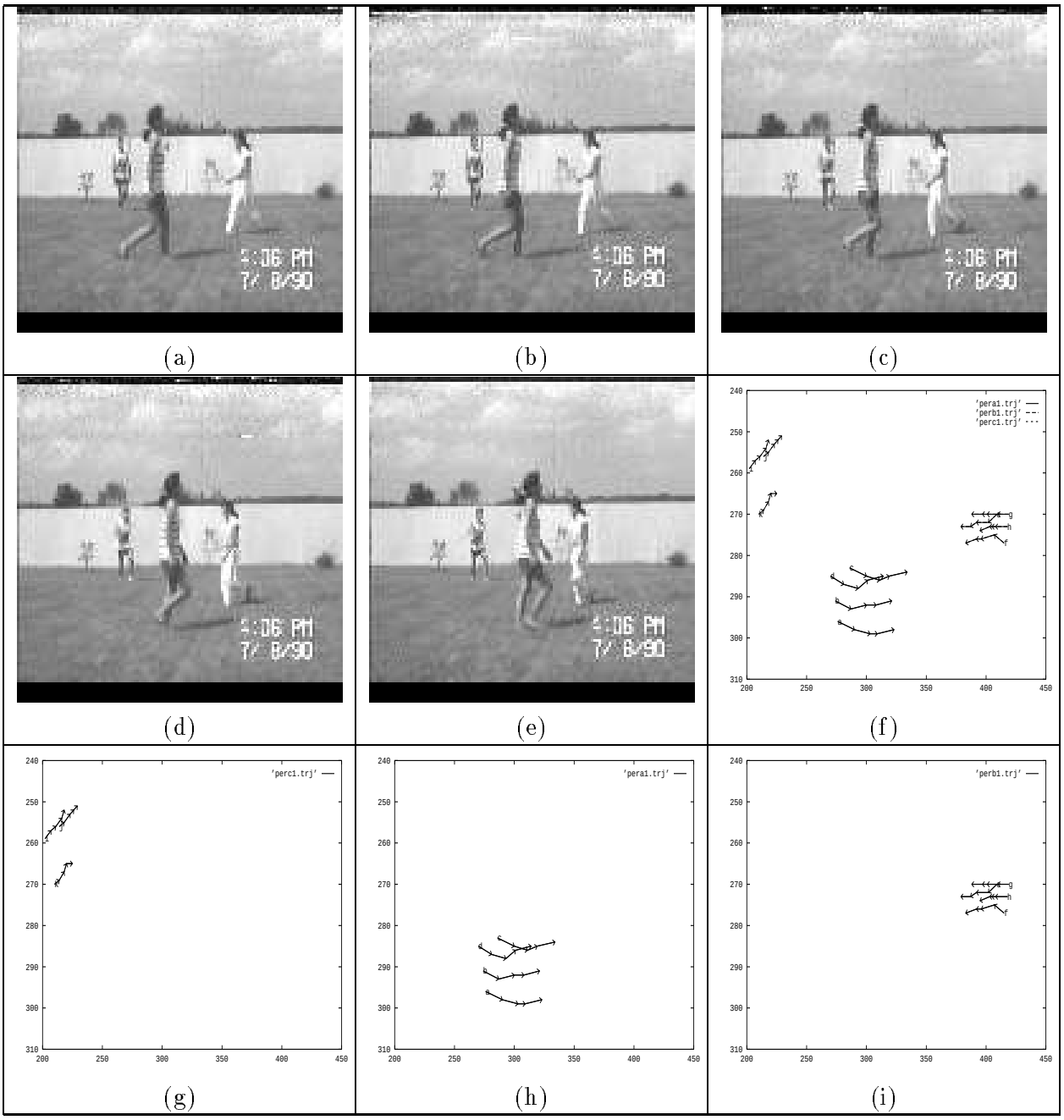

Figure 10: Results of segmentation algorithm for Walking sequence. (a)-(e) A sequence of a real scene with three walking persons: Female-1, Female-2, and Male-1. Points a, b, c and d on Female-1, points e, f, g, h on Female-2, and points i, j, and k on Male-1 were tracked. The proposed algorithm came up with the correct segmentation. (f) The trajectories of the marked points over 5 frames. (g) The group of trajectories belonging to Female-1. (h) The group of trajectories belonging to Female-2. (i) The group of trajectories belonging to Male-1.

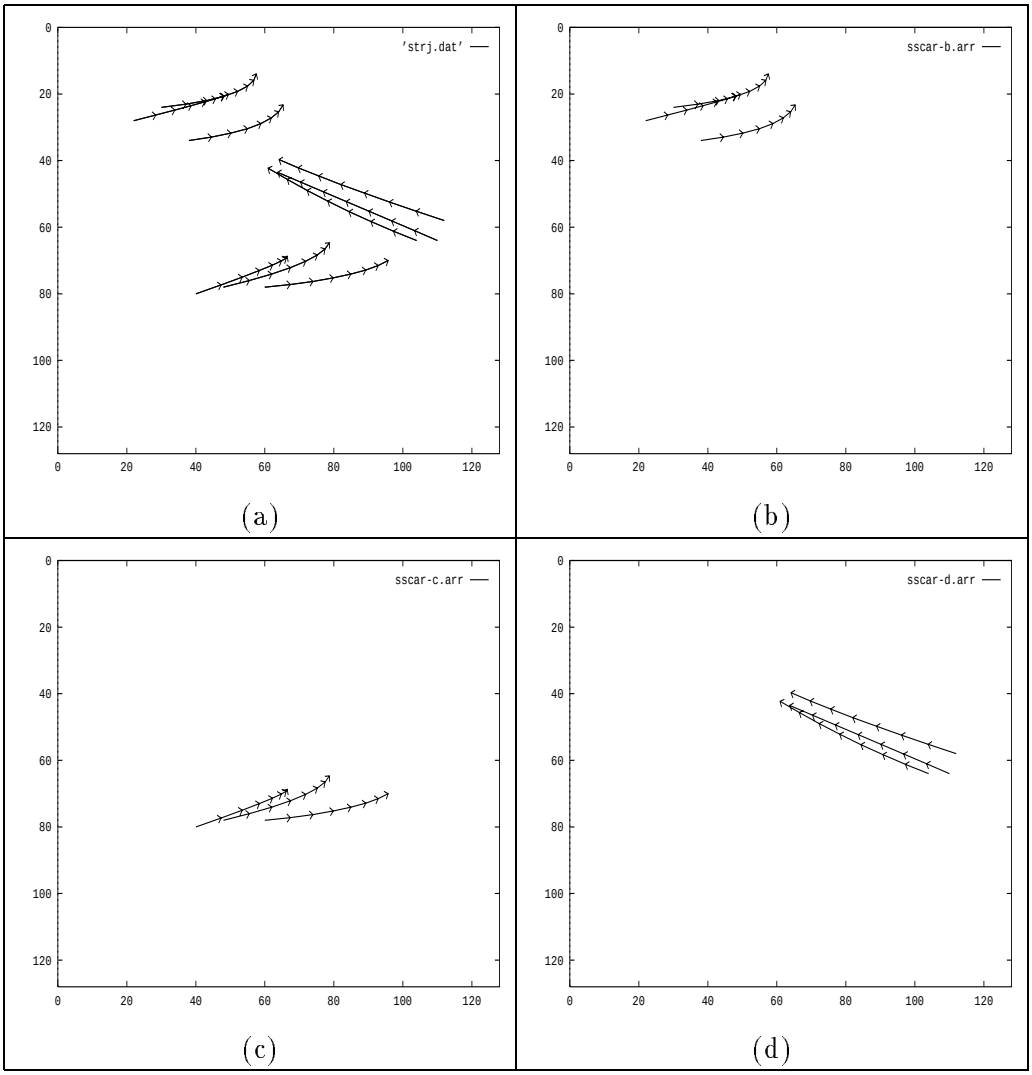

Figure 11: Results of segmentation algorithm for the Car sequence. (a) The trajectories generated in Figure 2(i). (b) The group of trajectories belonging to Jeep. (c) The group of trajectories belonging to Truck. (d) The group of trajectories belonging to Car.

Black Box and points l, m, n and o on Metal Box were tracked. The camera was kept at a larger distance from the objects as compared to the the previous experiment. Our algorithm came up with the correct segmentation. In this sequence, there are 15 trajectories. The total number of possible 3-combinations is $\binom{15}{3} = 455$, but only 24 survived the first step. The number of 4-combinations out of this set was 5. There were no combinations of 5 or higher. The final segmentation had three 4-combinations and one 3-combination. In both experiments, we found that the constraint that two of the three angles of the FOE triangle should be small was itself sufficient for the correct segmentation. However, the distance and the sign constraint were also found to be satisfied in both experiments, even though the scene would have been segmented correctly without them. Figure 9(k) shows some sample numerical values of the terms computed in steps 1(b) and 1(c) of the segmentation algorithm using FOE triangle.

Previous examples of sequences of real scenes were generated in the laboratory with objects moved by hand. We also performed some experiments on a sequence of real scenes with natural

motion. The trajectories generated in Figure 2(h) for the Car Sequence were inputed to our algorithm for segmentation, and correct segmentation was obtained. The results are shown in Figure 11. In the next experiment, three persons, Female-1, Female-2 and Male-1, walking in a backyard were videotaped. Female-1 walks from left to right, Female-2 walks from right to left but at different distances from the camera, and Male-1 walks diagonally from the upper left corner of the image to the bottom right corner. The male person walks slower than the females. Five frames from that sequence were digitized, and four points corresponding to each female, and three points corresponding to the male person were tracked. Our program also performed quite well on this data, and segmented the trajectories into three disjoint groups. The results are shown in Figure 10.

6 Conclusion

This paper deals with the generation and segmentation of motion trajectories, which is crucial to structure from motion approaches. In contrast to the conventional approaches for generation of motion trajectories using motion correspondence, we employ sequences of optical flows, and link the flow vectors for selected points into motion trajectories. We also present a simple algorithm for segmenting motion trajectories into groups belonging to individual moving objects. Our approach uses some useful properties of the Focus of Expansion, however, it does not rely on the precise location of FOE. Both algorithms have been tested against a large number of sequences. In all cases the results are very encouraging. Our method uses very few parameters (one or two); the values of these parameters were almost constant for all the experiments.

Acknowledgements: The authors are thankful to Professor Allen Hanson of University of Massachsetts for providing us the image sequences and Anandan's optical flow program.

References

[1] Adiv, G. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 7:384–401, 1985.

[2] Anandan, P. *A Computational Framework and an Algorithm for Measurement of Visual Motion.* International Journal of Computer Vision, 2, 283-310, 1989.

[3] Ayache, N. *Artificial Vision for Mobile Robots* MIT Press, 1991.

[4] Barron, J.L., Fleet, D.J., Beauchemin, S.S. and Burkitt, T.A. Performance of Optical Flow Techniques. In Proc. of Computer Vision and Pattern Recognition. pages 236-242, 1992.

[5] Boldt, M. and Weiss, R. Token-based extraction of straight lines. Technical Report 87-104, University of Massachusetts at Amherst, 1988.

[6] D.J. Heeger. A model for the extraction of image flow. *International Conference on Computer Vision* , pages 181–190, 1987.

[7] Kenner, M., and Pong, T. Motion analysis of long image sequence flow. *Pattern Recognition Letters*, 11:123–131, 1990.

[8] Moravec, H.P. Towards automatic visual obstacle avoidance. In *Proceedings of International Joint Conference on Artificial Intelligence-5*, page 584, MIT, Cambridge, Masachussets, 1977.

[9] Price, K.. Multi-frame feature based motion analysis. In *Proc. 10th Int Conf on Pattern Recognition*, pages 114–118. IEEE Computer Society, June 1990.

[10] Rangarajan, K., and Shah, M. Establishing Motion Correspondence. *CVGIP:Image Understanding*, 56-73, July 1991.

[11] Sawhney, H.S. and Oliensis, J.. Description and interpretation of rotational motion from image trajectories. DARPA Image Understanding workshop, pages 992–1003, 1989.

[12] Sethi, I.K., and Jain, R. Finding trajectories of Feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:56-73, 1987.

[13] Smith P.N., Sridhar B., and Hussien, B. Vision-Based Range Estimation Using Helicopter Flight Data. In Proc. of Computer Vision and Pattern Recognition. pages 202-208, 1992.

[14] Williams, L.R., and Hanson, A.R. Translating optical flow into token matches and depth from looming. In *Second International Conference on Computer Vision*, pages 441–448, Tampa, FL, December, 1988.