Tracking a Dynamic Set of Feature Points

Yi-Sheng Yao and Rama Chellappa, *Fellow, IEEE*

Abstract—**We address the problems of tracking a set of feature points over a long sequence of monocular images as well as how to include and track new feature points detected in successive frames. Due to the 3-D movement of the camera, different parts of the images exhibit different image motion. Tracking discrete features can therefore be decomposed into several independent and local problems. Accordingly, we propose a *localized* feature tracking algorithm. The trajectory of each feature point is described by a 2-D kinematic model. Then to track a feature point, an interframe motion estimation scheme is designed to obtain the estimates of interframe motion parameters. Subsequently, using the estimates of motion parameters, corresponding points are identified to subpixel accuracy. Afterwards, the temporal information is processed to facilitate the tracking scheme. Since different feature points are tracked independently, the algorithm is able to handle the image motion arising from general 3-D camera movements. On the other hand, in addition to tracking feature points detected at the beginning, an efficient way to dynamically include new points extracted in subsequent frames is devised so that the information in a sequence is preserved. Experimental results for several image sequences are also reported.**

I. Introduction

EGO-MOTION estimation has been an important topic in image sequence analysis for more than a decade. Based on matches of a few discrete features such as points and lines over two or three frames, many algorithms for estimating the ego-motion of the camera and the structure of discrete features have been proposed. Although linear algorithms result when two or three frames are used, the high sensitivity of the estimates to input errors has been observed [1], [10]. Meanwhile, the robustness of approaches which use a sequence of images has attracted the attention of many researchers [15], [17]. The issue of finding feature correspondences over a long sequence of images needs to be addressed in such approaches.

Existing techniques for tracking a set of discrete features over a sequence of images generally fall into two categories: two-frame based and long-sequence based.

1) *Two-frame-based approaches*: In this category, finding feature correspondences over a sequence of images is broken into successive, yet *independent* problems of two-view matching. For example, in [14], Weng *et al.* used multiple attributes of each image point such as intensity, edgeness, and cornerness, which are invariant under rigid motion in the image plane along with a set of constraints to compute a dense displacement field and occlusion areas in two images. Cui *et al.* [8] then used an intensity-based cross-correlation method to refine the two-view matching results and obtain feature point correspondences over the sequence. In [19], Zheng and Chellappa first applied an image registration technique to compensate for the motion of the camera between two consecutive frames. The feature point correspondence problem is then solved by repeatedly identifying the corresponding points to subpixel accuracy using the correlation matching method.
2) *Long-sequence-based approaches*: In this category, smoothness constraints are employed to exploit the temporal information existing in the sequence. For example, assuming that the motion of an object does not change abruptly, Sethi and Jain [13] formulated the correspondence problem as an optimization problem. The trajectories of a set of feature points are obtained by searching for a set of trajectories each of which has maximal smoothness. Blostein and Huang [3] used multistage hypothesis testing (MHT) to detect small, moving objects in each image; a feature trajectory is determined by repeatedly detecting the same feature point over the sequence. Chang and Aggarwal [6] assumed a 2-D kinematic motion model and applied the joint probabilistic data association filter (JPDAF) to track line segments with the ability to initiate or terminate the trajectory of a line segment. Employing a 3-D kinematic motion model and a Mahalanobis distance-based matching criterion, Zhang and Faugeras [18] applied an extended Kalman filter (EKF) to track a set of line segments. A fading memory type statistical test was suggested to take into account the occlusion and disappearance of line segments.

In essence, when the motion of the camera is smooth such that the smoothness constraints hold, long-sequence-based approaches are likely to outperform two-frame-based methods. On the contrary, if the movements of the camera between two frames vary often in the sequence and results in nonsmooth image motion, two-frame-based schemes seem to capture the variations more promptly.

In this paper, the merits of both long-sequence and two-frame-based methods are considered in designing an algorithm for finding trajectories of a set of feature points over a sequence. Basically, for each feature point of interest, a *local*

Manuscript received July 5, 1994; revised January 12, 1995. This work was supported by the Advanced Research Projects Agency under Order 8459 and the U.S. Army Topographic Engineering Center under Contract DACA 76-92-C-0009. The associate editor coordinating the review of this paper and approving it for publication was Prof. A. Murat Tekalp.

The authors are with the Department of Electrical Engineering, Computer Vision Laboratory, University of Maryland, College Park, MD 20742-3275 USA.

IEEE Log Number 9413835.

2-D constant translational and rotational motion model is employed to exploit the temporal information in a sequence. On the other hand, to account for the nonsmooth image motion due to 3-D camera movements, a scheme is devised to estimate the interframe motion between two windows in which the corresponding points of the feature point of interest are likely to appear. Consequently, in each frame, the temporal information up to that particular frame is explicitly stored in a state vector; the state vector consists of the position of the corresponding point in that particular frame as well as the motion parameters between the previous frame. Since the state vector in each frame contains the accumulated information, tracking a feature point over a sequence is therefore decomposed into successive two-frame matching problems. However, different from two-frame-based approaches, the resulting problems are not independent.

Subsequently, to find the corresponding point of a feature point in the consecutive frame, a tracking algorithm containing three stages are performed: interframe motion, corresponding point identification and temporal information filtering. The first stage uses a Probabilistic Data Association Filter (PDAF) to obtain the estimates of 2-D motion parameters between two frames. The next stage employs techniques such as image warping, correlation matching, image differential estimation and bilinear interpolation to identify the corresponding point in the frame of interest. Finally, the last stage uses an EKF to update the state vector so that new temporal information can be processed.

In addition, when a sequence of images is considered, the scene in the images changes constantly. Feature points detected from the subsequent frames are thereupon likely to provide more information for other higher-level applications such as obstacle avoidance, time-to-collision, etc. A scheme is thus designed to include these newly extracted feature points. Accordingly, the algorithm has the ability to track a dynamic set of feature points.

Our approach differs from currently existing discrete feature tracking techniques in various aspects:

1) The algorithm has the merits of both two-frame and long-sequence-based approaches.
2) Tracking a set of feature points is decomposed *spatially* into independent tasks: Except for the 3-D motion-based algorithms such as [18], we feel that tracking a feature point is a local problem whenever 2-D image motion models are employed. This contrasts our work with the approach reported in [19]. As shown in the experiments, by using a localized tracking scheme, simple 2-D motion models can handle the image motion due to general 3-D movements of the camera.
3) The inclusion of new feature points is addressed: The algorithm includes a scheme for tracking new feature points detected from subsequent frames. The scheme is efficient in the sense that only a limited number of feature points are tracked at all times.
4) The imperfectness of feature point extraction algorithms is considered: Many feature tracking algorithms assume two steps in tracking discrete features. The first step is to detect features using a feature detection scheme. Subsequently, different algorithms employ different methods in matching a set of currently tracked features to the set of features detected in the previous step. The accuracy of these algorithms thereupon relies on the feature detection schemes [6], [13], [18]. However, detecting the same features in subsequent frames is not easy. On the other hand, our approach identifies corresponding points using other techniques. The outputs from a feature detection scheme are only used in the stage of interframe motion. Thus, better performance is expected.

Finally, before we describe the tracking algorithm, we would like to point out the limitation of this scheme. The algorithm is explicitly designed to establish feature point trajectories over an image sequence such that the full or partial knowledge regarding the ego-motion of the camera can be recovered later. Since a correlation-type matching method is employed to identify the corresponding points in subsequent frames, the algorithm is therefore unable to track points on the boundary of moving objects which move independently from the camera.

The organization of this paper is as follows. Next section gives the trajectory model employed in this work. Section III presents the algorithm for finding corresponding points between two frames. A scheme for including new feature points is described in Section IV. Experimental results are reported in Section V and conclusions are given in Section VI.

II. Trajectory Model

There are, in general, two ways to describe the motion between two frames: 3-D-based and 2-D-based methods. Three-dimensional-based methods resort to the understanding of the camera motion while 2-D-based approaches apply 2-D transformations. For the 3-D-based methods, since the motion of the camera needs to be estimated from the images, finding corresponding points is then mixed with motion/structure estimation schemes. Because of the coupling of these two stages, errors in one stage are likely to affect the accuracy in the other one. Besides, there are applications in which the motion information is not required while finding matching points is important [5]. It is therefore desirable to have a method independent of the 3-D motion estimation scheme. Accordingly, 2-D-based approaches are considered.

Among 2-D-based techniques, three transformations: Affine, projective and polynomial, are most commonly employed in a closely related problem, namely, image registration [4], [16]. Their appropriateness depends on the underlying motion of the camera. For convenience, denote $\boldsymbol{r}_k$: $(x(k), y(k))$ as a point in the kth frame and $\boldsymbol{r}_{k+1}$: $(x(k+1), y(k+1))$ as the resulting image point in the $(k+1)$th frame. The affine transformations account for rotation, translation, scaling and skew between two images as follows:

$$\boldsymbol{r}_{k+1} = \boldsymbol{A}_k \boldsymbol{r}_k + \boldsymbol{t}_k \tag{1}$$

where $\boldsymbol{A}_k$ and $\boldsymbol{t}_k$ are the transformation matrix and vector. Although affine transformations are easy to use, perspective distortions between images are not considered. To capture more closely the image motion, the projective transformations

can be used. They are expressed as

$$x(k+1) = \frac{a_k x(k) + b_k y(k) + c_k}{d_k x(k) + e_k y(k) + 1} \quad (2)$$

$$y(k+1) = \frac{f_k x(k) + g_k y(k) + h_k}{d_k x(k) + e_k y(k) + 1} \quad (3)$$

where $(a_k, b_k, \cdots, h_k)$ are the transformation coefficients. Nonetheless, there still exist distortions not accounted by affine and projective transformations [4], [16]. If higher satisfactory results are desired, nonlinear transformations such as polynomial transformations can be used:

$$x(k+1) = \sum_{i=0}^{N_k} \sum_{j=0}^{N_k - i} a_{kij} x(k)^i y(k)^j \quad (4)$$

$$y(k+1) = \sum_{i=0}^{N_k} \sum_{j=0}^{N_k - i} b_{kij} x(k)^i y(k)^j \quad (5)$$

where N_k is the order of the transformation. (a_{kij}, b_{kij}) are the constant polynomial coefficients.

While the goal of image registration is to match two pictures *globally*, finding corresponding points of a feature point is, in essence, a *local* problem. The locality results from the perspective distortion as well as the camera motion. In this work, for a feature point of interest, a transformation is chosen to describe the image motion between two windows in which the matching points are likely to appear. Since only two windows are concerned, various distortions between the two windows are expected to be small. An affine transformation is therefore used. Moreover, to exploit the temporal information of a sequence, constant translational and rotational dynamics are assumed. Together, the image motion of the jth feature point, between the kth and $(k+1)$th frames, is modeled as

$$\boldsymbol{x}_j(k+1) = \boldsymbol{f}[\boldsymbol{x}_j(k)] + \boldsymbol{w}_j(k+1) \quad (6)$$

with

$$\boldsymbol{x}_j(k) = [x_j(k) \quad y_j(k) \quad t_{jx}(k) \quad t_{jy}(k) \quad \theta_j(k)]^T \quad (7)$$

and

$$\boldsymbol{f}[\boldsymbol{x}_j(k)] = \begin{pmatrix} x_j(k)\cos\theta_j(k) - y_j(k)\sin\theta_j(k) + t_{jx}(k) \\ x_j(k)\sin\theta_j(k) + y_j(k)\cos\theta_j(k) + t_{jy}(k) \\ t_{jx}(k) \\ t_j y(k) \\ \theta_j(k) \end{pmatrix} \quad (8)$$

where $(x_j(k), y_j(k))$ is the position of the corresponding point in the kth frame. $t_{jx}(k), t_{jy}(k), \theta_j(k)$ respectively denote the associated translational movement along the x, y directions, the rotation angle, all between the $(k-1)$th image and the kth image; they are referred to as the motion parameters between the $(k-1)$th frame and the kth frame. $\boldsymbol{w}_j(\cdot)$ is zero mean, white noise which is added to account for the deviation of the model from the stated assumption.

As argued above, the consideration of a localized tracking scheme results in the choice of the simple affine transformation in this work. As shown in the experiments later, this model more or less captures the local image motion arising from arbitrary camera movements. It is therefore seen as a transformation between the 3-D-based tracking methods and the 2-D global image registration techniques.

III. Feature Tracking Between Two Frames

As mentioned earlier, tracking feature points over a sequence is decomposed temporally into successive two-frame matching problems. Therefore, after the trajectory model has been chosen, we describe the tracking algorithm between two frames in this section. Without loss of generality, assuming that the jth feature point has been tracked up to the kth frame, the algorithm is described in terms of extending the trajectory to the $(k+1)$th frame. For clarity, an overview of the algorithm is first given and each step of the algorithm is then described successively.

A. Overview of the Algorithm

To design an algorithm that combines the merits of both long-sequence and two-frame-based approaches, two issues are considered in devising a tracking scheme: How to exploit the temporal information such that the search area containing the corresponding point is small, and how to identify the corresponding point in the $(k+1)$th frame. In our work, an EKF/PDAF is employed to process the temporal information while a correlation-matching-based technique identifies the corresponding point to subpixel accuracy. The resulting scheme therefore involves three stages: Interframe motion estimation, corresponding point identification and temporal information filtering.

More specifically, the two-frame tracking algorithm contains several steps that are summarized in the following:

- **Interframe motion estimation**: In the first stage, an interframe motion estimation scheme is applied. The scheme uses the accumulated temporal information up to the kth frame to obtain an estimate of the motion between two neighborhoods which are likely to contain corresponding points of the jth feature point.
- **Corresponding point identification**: The second stage identifies the corresponding point of the jth feature point in the $(k+1)$th frame by the following procedures:
  a) *Forward window warping and window extraction*: Based on the interframe motion parameters, this step predicts the position of the corresponding point in the $(k+1)$th frame. An image warping technique is applied to generate a window centered at the predicted location from a neighborhood of the kth-frame corresponding point. Also, another window possibly containing the corresponding point is extracted from the $(k+1)$th frame.
  b) *Grid neighbors matching*: The second step employs a correlation-type matching method to find the corresponding point. However, the predicted location does not necessarily fall onto a grid location in the generated window. Alternatively, matching points of the four nearest grid neighbors are identified here.

c) *Correct match verification*: Before doing any further process, if the corresponding point is less likely to be reliably identified based on the matches from the previous step, the algorithm stops tracking the feature point in this step.

d) *Subpixel accuracy refinement*: Once the algorithm thinks the corresponding point can be reliably found in the $(k+1)$th frame, this step applies an image-differential-based scheme to improve the correlation matching results.

e) *Matching point identification*: Finally, in the last step, the algorithm uses a bilinear interpolation scheme to obtain the corresponding point in the $(k+1)$th frame.

- **Temporal information filtering**: The third stage processes the temporal information contained in the $(k+1)$th frame by updating the state vector $\boldsymbol{x}_j(k+1)$.

After these procedures are completed, the algorithm can continue tracking the jth feature point to the $(k+2)$th frame. We now describe each step of the two-frame feature tracking algorithm in the following.

B. Interframe Motion Estimation

Temporal information contained in a sequence, in general, can be used to facilitate the feature tracking problem. To exploit the temporal information and accordingly, reduce the search area for finding the corresponding point, an interframe motion estimation scheme is devised. In this work, the PDAF [2] which was originally proposed for tracking a moving object in a cluttered environment is applied for this step. A cluttered environment, in the context of target tracking, is referred to the situation that there exist ambiguities in observations such that the designed tracking scheme does not know which observation is correct, or whether it is from the tracked object. Similarly, for tracking the jth feature point, the ambiguities appear whenever there are more than one possible matching candidates in the $(k+1)$th frame.

For convenience, in addition to the trajectory model (6), define an observation model which relates the state vector $\boldsymbol{x}_j(k+1)$ with the noisy, but correct observation $\boldsymbol{z}_j(k+1)$ as follows:

$$\boldsymbol{z}_j(k+1) = \boldsymbol{H}\boldsymbol{x}_j(k+1) + \boldsymbol{n}_j(k+1) \tag{9}$$

where $n_j(\cdot)$ is the zero mean, white observation noise

$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}. \tag{10}$$

Based on the estimates of the state vector at the kth frame $\hat{\boldsymbol{x}}_j(k|k)$

$$\hat{\boldsymbol{x}}_j(k|k) = [\hat{x}_j(k|k) \;\; \hat{y}_j(k|k) \;\; \hat{t}_{jx}(k|k)\hat{t}_{jy}(k|k) \;\; \hat{\theta}_j(k|k)]^T \tag{11}$$

the interframe motion estimation scheme first predicts the $(k+1)$th-frame state vector, $\check{\boldsymbol{x}}_j(k+1|k)$, according to (6)

$$\check{\boldsymbol{x}}_j(k+1|k) = \boldsymbol{f}[\hat{\boldsymbol{x}}_j(k|k)]. \tag{12}$$

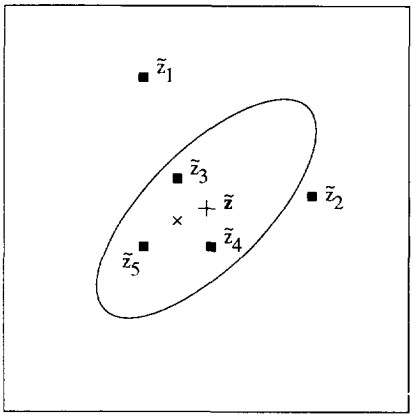

Fig. 1. Illustrations of the PDAF: $\tilde{z}$ is the predicted location and the ellipse represents the associated validation gate. $\{\tilde{z}_i, i = 1, \cdots, 5\}$ are the five extracted feature points. Among them, $\{\tilde{z}_3, \tilde{z}_4, \tilde{z}_5\}$ are inside the validation gate. The position marked by $\times$ corresponds to the position estimates computed by the PDAF.

In particular, the predicted location of the corresponding point, $\tilde{\boldsymbol{z}}_j(k+1|k) = (\tilde{x}_j(k+1|k), \tilde{y}_j(k+1|k))$ is obtained as

$$\begin{aligned} \tilde{x}_j(k+1|k) &= \cos(\hat{\theta}_j(k|k))\hat{x}_j(k|k) \\ &\quad - \sin(\hat{\theta}_j(k|k))\hat{y}_j(k|k) + \hat{t}_{jx}(k|k) \\ \tilde{y}_j(k+1|k) &= \sin(\hat{\theta}_j(k|k))\hat{x}_j(k|k) \\ &\quad + \cos(\hat{\theta}_j(k|k))\hat{y}_j(k|k) + \hat{t}_{jy}(k|k). \end{aligned} \tag{13}$$

Subsequently, a window centered at $\tilde{\boldsymbol{z}}_j(k+1|k)$ is extracted from the $(k+1)$th frame and the feature point extraction algorithm reported in [12] is applied to the window to identify salient feature points. Due to the lack of information as well as the desire for higher accuracy, the corresponding point is not identified from the extracted points. Alternatively, a validation gate based on the Mahalanobis distance [7], [9] is constructed to select points for further processing. More specifically, define a validation gate centered at $\tilde{\boldsymbol{z}}_j(k+1|k)$ and with parameter γ as [2], [7]:

$$\begin{aligned} \boldsymbol{V}_{j,k+1}(\gamma) = \{\boldsymbol{z}: &[\boldsymbol{z} - \tilde{\boldsymbol{z}}_j(k+1|k)]^T \boldsymbol{S}_j^{-1}(k+1) \\ &\cdot [\boldsymbol{z} - \tilde{\boldsymbol{z}}_j(k+1|k)] \le \gamma\} \end{aligned} \tag{14}$$

where $\boldsymbol{S}_j(k+1)$ is the covariance matrix of the innovation vector $\boldsymbol{z} - \tilde{\boldsymbol{z}}_j(k+1|k)$, and γ decides the scope of the validation gate. A set of extracted points is selected if their Mahalanobis distances are less than γ. Without loss of generality, it is assumed that there are $m_j(k+1)$ points, denoted as $\{\tilde{\boldsymbol{z}}_{j,i}(k+1), i = 1, \cdots, m_j(k+1)\}$, inside the validation gate $\boldsymbol{V}_{j,k+1}(\gamma)$. For clarity, Fig. 1 shows a situation where five points have been extracted, the three points inside the validation gate will be selected.

Among the $m_j(k+1)$ extracted points, due to changes between images, each of the $m_j(k+1)$ points can be the corresponding point. In other words, if we consider these points as *noisy* corresponding points, we face a situation with ambiguities in observations. To process the information contained in these noisy observations, the PDAF is thereafter employed.

In essence, the PDAF obtains the information by associating different weights to different observations. A weight

is assigned to an observation according to the *a posteriori* probability of an event which states that the corresponding observation is the correct one given the past data. As shown in [2], under the assumption that the observations are Normally distributed about $\tilde{\boldsymbol{z}}_j(k+1|k)$, these weights can be easily obtained. For convenience, denote these weights by $\{\beta_{j,i}(k+1), i = 0, 1, \cdots, m_j(k+1)\}$ respectively. Note that $\beta_{j,0}(k+1)$ is the weight assigned to the event that none of the observations are correct.

After each weight has been computed, the PDAF (or the interframe motion estimation scheme) processes the information from the $m_j(k+1)$ observations as follows

$$\tilde{\boldsymbol{x}}_j(k+1|k+1) = \tilde{\boldsymbol{x}}_j(k+1|k) + \tilde{\boldsymbol{K}}_j(k+1) \cdot \left[\sum_{i=1}^{m_j(k+1)} \beta_{j,i}(k+1)\tilde{\boldsymbol{v}}_{j,i}(k+1)\right] \tag{15}$$

where $\tilde{\boldsymbol{K}}_j(k+1)$ is the gain matrix defined similarly as the EKF

$$\tilde{\boldsymbol{v}}_{j,i}(k+1) = \tilde{\boldsymbol{z}}_{j,i}(k+1) - \tilde{\boldsymbol{z}}_j(k+1|k). \tag{16}$$

For illustration purpose, the first two components of the state vector $\tilde{\boldsymbol{x}}_j(k+1|k+1), (\tilde{x}_j(k+1|k+1), \tilde{y}_j(k+1|k+1))$, are supposedly marked by $\times$ in Fig. 1. It is worth noting that if additional information is available such that the ambiguities in observations are resolved, then a PDAF is simplified to an EKF. This justifies the state vector prediction in (12).

Consequently, from (15), the estimates of motion parameters, i.e., the other three elements in $\tilde{\boldsymbol{x}}_j(k+1|k+1)$: $\tilde{t}_{jx}(k+1|k+1), \tilde{t}_{jy}(k+1|k+1), \tilde{\theta}_j(k+1|k+1)$, are obtained. We refer to them as the interframe motion parameters. Since the information in the $(k+1)$th frame is incorporated, the interframe motion parameters more or less capture the image motion of the neighborhoods of the jth feature point due to the 3-D movement of the camera. With these parameters available, the algorithm proceeds to the next stage to identify the corresponding point.

C. Corresponding Point Identification

In this stage, the corresponding point of the jth feature point in the $(k+1)$th frame is found using a correlation matching approach followed by an interpolation scheme. The employed correlation matching method is similar to the block matching technique; however, interframe motion parameters are used so that the search area of the corresponding point is small and the matching results are more accurate. In addition, the interpolation scheme handles the problem due to the corresponding point not being at a grid location. This is important for tracking a feature point over a long sequence. Approximating the corresponding point by its nearest grid neighbor often leads to the situation that the corresponding point slowly moves away from the right position. The procedures used in this stage are presented in the following.

1) Forward Window Warping and Window Extraction: The first step in identifying the corresponding point consists of obtaining two windows from the kth and $(k+1)$th frames respectively. We refer to the window obtained from the kth frame as the reference window $I_{j,r}$. The other one is called the target window $I_{j,t}$.

Consider $I_{j,r}$ first. Given the kth frame and interframe motion parameters, we want to *predict* the reference window $I_{j,r}$. Similar problems have been studied in the area of image warping [16]. Basically, $I_{j,r}$ is an image that the algorithm believes the neighborhood of the corresponding point in the $(k+1)$th frame should look like. To obtain $I_{j,r}$, a more accurate prediction regarding the position of the corresponding point, $\hat{\boldsymbol{z}}_{j,r}(k+1|k) = (\hat{x}_{j,r}(k+1|k), \hat{y}_{j,r}(k+1|k))$, is used:

$$\begin{aligned}
\hat{x}_{j,r}(k+1|k) = {} & \cos(\tilde{\theta}_j(k+1|k+1))\hat{x}_j(k) \\
& - \sin(\tilde{\theta}_j(k+1|k+1))\hat{y}_j(k) \\
& + \tilde{t}_{jx}(k+1|k+1) \\
\hat{y}_{j,r}(k+1|k) = {} & \sin(\tilde{\theta}_j(k+1|k+1))\hat{x}_j(k) \\
& + \cos(\tilde{\theta}_j(k+1|k+1))\hat{y}_j(k) \\
& + \tilde{t}_{jy}(k+1|k+1)
\end{aligned} \tag{17}$$

where $\hat{\boldsymbol{z}}_j(k) = (\hat{x}_j(k), \hat{y}_j(k))$ is the position of the previously found kth frame corresponding point. Note that $(\hat{x}_j(k), \hat{y}_j(k))$ is used instead of $(\hat{x}_j(k|k), \hat{y}_j(k|k))$. Subsequently, assume that pixels which are close to $(\hat{x}_j(k), \hat{y}_j(k))$ in the kth frame undergo the same interframe motion, their locations in the $(k+1)$th frame are predicted in a similar way. After that, the predicted pixels are assigned with the same intensity values as the original pixels. This yields the reference window $I_{j,r}$ whose center is $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$.

Next, by centering at $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$, $I_{j,t}$ is directly extracted from the $(k+1)$th frame. To avoid potential confusion, we denote the center of $I_{j,t}$ as $\hat{\boldsymbol{z}}_{j,t}(k+1|k) = (\hat{x}_{j,t}(k+1|k), \hat{y}_{j,t}(k+1|k))$, although $\hat{\boldsymbol{z}}_{j,t}(k+1|k)$ has the same value as $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$.

Since $I_{j,r}$ is obtained with the knowledge of the interframe motion, a correlation matching method can be employed to find the corresponding point. Thus, given $I_{j,r}$ and $I_{j,t}$, if $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$ happens to be a grid point, then the corresponding point of the jth feature point can be directly found. However $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$, in general, is not located at a grid point. We describe how to handle this situation next.

2) Grid Neighbors Matching: Correlation matching techniques typically match a grid point in one image with a grid point in another image. In our case, the problem appears when the predicted location $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$ does not coincide with a grid point. A scheme suggested in [19] is employed to solve this problem. The approach is composed of two steps:

1) Find matching points, in $I_{j,t}$, of the four nearest grid points of $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$ using correlation matching methods.
2) Interpolate the results to obtain the desired corresponding point. We describe the first step here. The second step is discussed later.

For convenience, denote the four nearest grid points of $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$ as $\{\hat{\boldsymbol{z}}_{j,i}(k+1|k), i=1,\cdots,4\}$

$$\begin{aligned}
\hat{\boldsymbol{z}}_{j,1}(k+1|k) &= ([\hat{x}_{j,r}(k+1|k)], [\hat{y}_{j,r}(k+1|k)]) \\
\hat{\boldsymbol{z}}_{j,2}(k+1|k) &= ([\hat{x}_{j,r}(k+1|k)], [\hat{y}_{j,r}(k+1|k)]+1) \\
\hat{\boldsymbol{z}}_{j,3}(k+1|k) &= ([\hat{x}_{j,r}(k+1|k)]+1, [\hat{y}_{j,r}(k+1|k)]) \\
\hat{\boldsymbol{z}}_{j,4}(k+1|k) &= ([\hat{x}_{j,r}(k+1|k)]+1, [\hat{y}_{j,r}(k+1|k)]+1)
\end{aligned} \tag{18}$$

where $[\cdot]$ represents the floor function. Without loss of generality, we focus on finding the matching point of $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$ here. The matching points of the other three grid points can be found in similar fashions.

To employ the correlation matching method, a template $\Omega_{I_{j,r}}(\hat{\boldsymbol{z}}_{j,1}(k+1|k))$ centered at $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$ is first created from $I_{j,r}$. Then, in $\Omega_{I_{j,r}}(\hat{\boldsymbol{z}}_{j,1}(k+1|k))$, to put more emphasis on pixels close to $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$, each pixel is further assigned with a different weight according to the following:

$$\mu_{l,m} = \begin{cases} 1, & \text{if } (l,m)=(0,0) \\ \dfrac{c}{8\ \max(|l|,|m|)}, & \text{otherwise} \end{cases} \tag{19}$$

where c is a constant and (l,m) are integers such that $([\hat{x}_{j,r}(k+1|k)]+l, [\hat{y}_{j,r}(k+1|k)]+m) \in \Omega_{I_{j,r}}(\hat{\boldsymbol{z}}_{j,1}(k+1|k))$. The resulting template is thereupon seen as a weighted template. Note that for any $\boldsymbol{q} \in I_{j,t}$, a template $\Omega_{I_{j,t}}(\boldsymbol{q})$ centered at $\boldsymbol{q}$ with similar weight assignments can also be obtained.

Subsequently, if we employ the weighted templates and define the similarity measure between $\boldsymbol{p}=(p_1,p_2) \in I_{j,r}$ and $\boldsymbol{q}=(q_1,q_2) \in I_{j,t}$ as seen at the bottom of the page in (20), where $g_1(\cdot), g_2(\cdot)$ are the intensity functions of pixels in $I_{j,r}$ and $I_{j,t}$ respectively, and

$$\overline{g}_1(\boldsymbol{p}) = \frac{1}{N}\sum_{l,m} g_1(p_1+l, p_2+m) \tag{21}$$

$$\overline{g}_2(\boldsymbol{q}) = \frac{1}{N}\sum_{l,m} g_2(q_1+l, q_2+m) \tag{22}$$

with N being the number of pixels in both $\Omega_{I_{j,r}}(\boldsymbol{p})$ and $\Omega_{I_{j,t}}(\boldsymbol{q})$, then the matching point of $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$ is obtained by searching for a grid point in $I_{j,t}$ which has the highest similarity measure with $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$. (For example, $-5 \le l,m \le 5$ and $c=2$ are used in our experiments.)

A few remarks regarding the above correlation matching scheme are given in the following. First, the similarity measure (20) has the same property as the well-known correlation coefficient [19]:

$$|\psi_{I_{j,r}} I_{j,t}| \le 1. \tag{23}$$

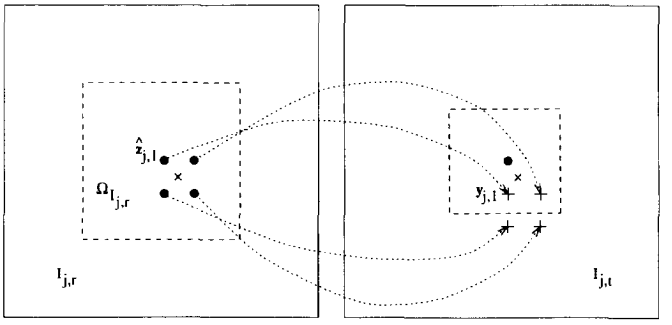

Fig. 2. Illustrations of grid neighbors matching: $\hat{\boldsymbol{z}}_{j,r}, \hat{\boldsymbol{z}}_{j,t}$ are marked by $\times$ in $I_{j,r}$ and $I_{j,t}$ respectively. The dashed square in $I_{j,r}$ represents the template $\Omega_{I_{j,r}}(\hat{\boldsymbol{z}}_{j,1})$. The dashed square in $I_{j,t}$ is the search area where the matching point of $\hat{\boldsymbol{z}}_{j,1}, \boldsymbol{y}_{j,1}$, is to be searched. The matching points of all four grid neighbors of $\hat{\boldsymbol{z}}_{j,r}$ are supposedly marked by $+$ in $I_{j,t}$.

Second, the use of weighted template is expected to achieve better localization because of the imperfect feature detection scheme. By this, we mean that the extracted feature points usually are not located exactly at the corners due to quantization effects. Typically, they are a few pixels away from the boundaries. Experiments show that by putting more weights to pixels near the center of the template helps achieving better localization later. Third, since the interframe motion has been considered, to find the matching grid point of $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$, only a small area centered at $([\hat{x}_{j,t}(k+1|k)], [\hat{y}_{j,t}(k+1|k)]+1)$ in $I_{j,t}$ needs to be searched. (In particular, a 10×10 window is searched in the experiments.) Fig. 2 illustrates a possible configuration obtained in this step. For convenience, we denote the resulting matching grid points of $\{\hat{\boldsymbol{z}}_{j,i}(k+1), i=1,\cdots,4\}$ as $\{\boldsymbol{y}_{j,i}(k+1), i=1,\cdots,4\}$.

Up to now, we have implicitly assumed that the corresponding point of the jth feature point exists in the $(k+1)$th frame, and the neighborhoods of corresponding points in the kth and $(k+1)$th frames are similar. However, there exist situations where these assumptions may not be valid. We discuss a scheme which verifies these assumptions next.

3) Correct Match Verification: It is well known that when tracking the jth feature point to the $(k+1)$th frame, the corresponding point itself is likely to be occluded. Even if the corresponding point were not occluded, the relative structures in the neighborhoods of corresponding points in the kth and $(k+1)$th frames may change dramatically. Both situations lead to incorrect matches using the previously designed correlation matching method. Therefore, before advancing to the next step, the hypothesis that the matches $\{(\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}_{j,i}(k+1)), i=1,\cdots,4\}$ are correct is tested here.

Accordingly, we select a threshold, say TH, to decide whether $(\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}_{j,i}(k+1))$ is a correct match for each

$$\psi_{I_{j,r}} I_{j,t}(\boldsymbol{p},\boldsymbol{q}) = \frac{\sum_{l,m} \mu_{lm}[g_1(p_1+l,p_2+m)-\overline{g}_1(\boldsymbol{p})][g_2(q_1+l,q_2+m)-\overline{g}_2(\boldsymbol{q})]}{\sqrt{\sum_{l,m}\mu_{lm}[g_1(p_1+l,p_2+m)-\overline{g}_1(\boldsymbol{p})]^2}\sqrt{\sum_{l,m}\mu_{lm}[g_2(q_1+l,q_2+m)-\overline{g}_2(\boldsymbol{q})]^2}} \tag{20}$$

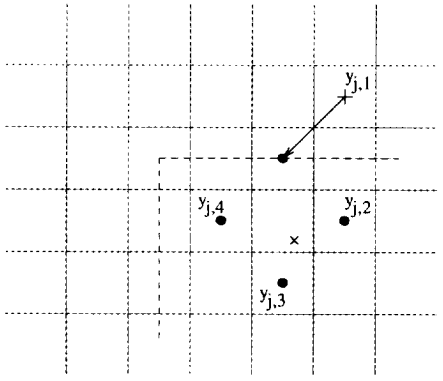

Fig. 3. Illustrations of Case 2 of correct match verification: × denotes the real corresponding point. The original $\boldsymbol{y}_{j,1}$ is denoted by + while the extrapolated one is indicated by •.

$i \in \{1, \cdots, 4\}$ as follows:

$$\begin{cases} \psi_{I_{j,r}} I_{j,t}(\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}_{j,i}(k+1)) \geq TH \\ \quad (\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}_{j,i}(k+1)) \text{ is correct} \\ \psi_{I_{j,r}} I_{j,t}(\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}_{j,i}(k+1)) < TH \\ \quad \text{otherwise.} \end{cases} \tag{24}$$

For convenience, denote the number of correct matches among $\{(\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}_{j,i}(k+1)), i = 1, \cdots, 4\}$ as n. Three cases are considered in the following:

Case 1: $n \leq 2$: In this case, more than two matches of grid neighbors of $\hat{\boldsymbol{z}}_{j,r}(k+1|k)$ are likely to be incorrect. Consequently, the algorithm feels that the real corresponding point of the jth feature point can not be reliably found in the $(k+1)$th frame. No further tracking will be attempted.

Case 2: $n = 3$: In this case, one of the matches is regarded as unreliable. Without loss of generality, we assume that $\boldsymbol{y}_{j,1}(k+1)$ is the unreliable matching point. Since there are three other correct matches, the algorithm tends to correct the wrong match using an extrapolation scheme. More specifically, because $\{\hat{\boldsymbol{z}}_{j,i}(k+1|k), i = 1, \cdots, 4\}$ form a square in $I_{j,r}$, the extrapolation scheme assumes that $\{\boldsymbol{y}_{j,i}(k+1), i = 1, \cdots, 4\}$ should form a parallelogram in $I_{j,t}$. The matching point of $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$ is then replaced by

$$\boldsymbol{y}_{j,1}(k+1) = \boldsymbol{y}_{j,2}(k+1) + \boldsymbol{y}_{j,3}(k+1) - \boldsymbol{y}_{j,4}(k+1). \tag{25}$$

For clarity, we illustrate this procedure in Fig. 3. In Fig. 3, assume that $\boldsymbol{y}_{j,1}(k+1)$ is originally matched to the conceivably wrong point marked by + due to changes in the neighborhood. Applying (25) yields another matching point for $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$ as indicated in the figure.

Case 3: $n = 4$: In this case, $\{(\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}_{j,i}(k+1)), i = 1, \cdots, 4\}$ are all confirmed to be correct.

Among three cases, if either Case 2 or Case 3 applies, the algorithm proceeds to the next step.

4) Subpixel Accuracy Refinement: Since $\{(\hat{\boldsymbol{z}}_{j,i}(k+1|k), i = 1, \cdots, 4\}$ are only matched to grid-level accuracy by the correlation matching scheme, this step intends to improve the matching accuracy to the subpixel level. In our work, an image-differential-based technique [11] is employed for the purpose [19]. Again, we illustrate this scheme in terms of $(\hat{\boldsymbol{z}}_{j,1}(k+1|k), \boldsymbol{y}_{j,1}(k+1))$.

Recall that $g_2(\cdot)$ is the intensity function of pixels in $I_{j,t}$. Under the assumption that $g_2(\cdot)$ has an offset $\boldsymbol{\Delta} = (\delta_x, \delta_y)$ relative to another intensity function $g_2'(\cdot)$ at $\boldsymbol{y}_{j,1}(k+1)$ as

$$g_2(\boldsymbol{y}_{j,1}(k+1)) = g_2'(\boldsymbol{y}_{j,1}(k+1) + \boldsymbol{\Delta}) \tag{26}$$

the image differential scheme approximates the difference between $g_2(\boldsymbol{y}_{j,1}(k+1))$ and $g_2'(\boldsymbol{y}_{j,1}(k+1)), d(\boldsymbol{y}_{j,1}(k+1))$, using the first-order Taylor series expansion:

$$d(\boldsymbol{y}_{j,1}(k+1)) \equiv g_2(\boldsymbol{y}_{j,1}(k+1)) - g_2'(\boldsymbol{y}_{j,1}(k+1)) \tag{27}$$

$$\approx \langle \nabla g_2(\boldsymbol{y}_{j,1}(k+1)), \boldsymbol{\Delta} \rangle \tag{28}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product of the arguments, and $\nabla g_2(\boldsymbol{y}_{j,1}(k+1))$ is the gradient vector

$$\nabla g_2(\boldsymbol{y}_{j,1}(k+1)) \equiv \left(\frac{\partial g_2(\boldsymbol{y}_{j,1}(k+1))}{\partial x}, \frac{\partial g_2(\boldsymbol{y}_{j,1}(k+1))}{\partial y} \right)^T. \tag{29}$$

Subsequently, assume that (28) holds for a small neighborhood around $\boldsymbol{y}_{j,1}(k+1)$ of size $(2\omega_d + 1) \times (2\omega_d + 1)$, then a set of equations can be found:

$$G\Delta = D \tag{30}$$

where

$$\begin{aligned} G = [&\nabla g_2(\boldsymbol{y}_{j,1}(k+1) + u_{-\omega_d,-\omega_d}), \\ &\cdots, \nabla g_2(\boldsymbol{y}_{j,1}(k+1) + u_{0,0}), \\ &\cdots, \nabla g_2(\boldsymbol{y}_{j,1}(k+1) + u_{\omega_d,\omega_d})]^T \end{aligned} \tag{31}$$

$$\begin{aligned} D = [&d(\boldsymbol{y}_{j,1}(k+1) + u_{-\omega_d,-\omega_d}), \\ &\cdots, d(\boldsymbol{y}_{j,1}(k+1) + u_{0,0}), \\ &\cdots, d(\boldsymbol{y}_{j,1}(k+1) + u_{\omega_d,\omega_d})]^T \end{aligned} \tag{32}$$

with

$$u_{\nu_1,\nu_2} = (\nu_1, \nu_2)^T, \quad -\omega_d \leq \nu_1, \nu_2 \leq \omega_d \tag{33}$$

such that $(\boldsymbol{y}_{j,1}(k+1) + u_{\nu_1,\nu_2})$ is in the $(2\omega_d+1) \times (2\omega_d+1)$ neighborhood of $\boldsymbol{y}_{j,1}(k+1)$.

Afterwards, the offset vector $\boldsymbol{\Delta}$ is obtained from (30) using a least-square approach. (In our experiments, $\omega_d = 3$.) $\hat{\boldsymbol{z}}_{j,1}(k+1|k)$ is then matched to $\boldsymbol{y}'_{j,1}(k+1) \equiv (x'_{j,1}(k+1), y'_{j,1}(k+1)) = (\boldsymbol{y}_{j,1}(k+1) + \boldsymbol{\Delta})$. Similarly, the subpixel matching of the other three grid points can also be obtained. We denote these four subpixel matching points as $\{\boldsymbol{y}'_{j,i}(k+1), i = 1, \cdots, 4\}$.

5) Matching Point Identification: After $\{\boldsymbol{y}'_{j,i}(k+1), i = 1, \cdots, 4\}$ have been obtained, the algorithm identifies the corresponding point of the jth feature point using a bilinear interpolation scheme in this step.

The scheme uses matches $\{(\hat{\boldsymbol{z}}_{j,i}(k+1|k), \boldsymbol{y}'_{j,i}(k+1)), i = 1, \cdots, 4\}$ to find the corresponding point of the jth feature point. More specifically, assume that for any pixel $\boldsymbol{p} = (p_1, p_2) \in I_{j,r}$ which is inside or on the boundary of the

square formed by $\{\hat{\boldsymbol{z}}_{j,i}(k+1|k), i=1,\cdots,4\}$, its matching point in $I_{j,t}, \boldsymbol{q}=(q_1,q_2)$, can be obtained as

$$\begin{aligned} q_1 &= \alpha_1 + \alpha_2 p_1 + \alpha_3 p_2 + \alpha_4 p_1 p_2 \\ q_2 &= \beta_1 + \beta_2 p_1 + \beta_3 p_2 + \beta_4 p_1 p_2 \end{aligned} \tag{34}$$

where $\{\alpha_i, \beta_i, i=1,\cdots,4\}$ are constant coefficients to be found.

Then, since the matching points of $\{\hat{\boldsymbol{z}}_{j,i}(k+1|k), i=1,\cdots,4\}$ have been obtained, $\{\alpha_i, \beta_i, i=1,\cdots,4\}$ can be found easily [16], [19]. Accordingly, using these coefficients and (34), the corresponding point of the jth feature point $\hat{\boldsymbol{z}}_j(k+1)=(\hat{x}_j(k+1), \hat{y}_j(k+1))$, are computed by

$$\begin{aligned} \hat{\boldsymbol{z}}_j(k+1) =& \boldsymbol{y}'_{j,1}(k+1) + \epsilon_x[\boldsymbol{y}'_{j,3}(k+1) \\ & - \boldsymbol{y}'_{j,1}(k+1)] + \epsilon_y[\boldsymbol{y}'_{j,2}(k+1) - \boldsymbol{y}'_{j,1}(k+1)] \\ & + \epsilon_x\epsilon_y[\boldsymbol{y}'_{j,1}(k+1) + \boldsymbol{y}'_{j,4}(k+1) \\ & - \boldsymbol{y}'_{j,2}(k+1) - \boldsymbol{y}'_{j,3}(k+1)] \\ \epsilon_x =& \hat{x}_{j,r}(k+1|k) - [\hat{x}_{j,r}(k+1|k)] \\ \epsilon_y =& \hat{y}_{j,r}(k+1|k) - [\hat{y}_{j,r}(k+1|k)] \end{aligned} \tag{35}$$

where $(\hat{x}_{j,r}(k+1|k), \hat{y}_{j,r}(k+1|k))$ and $([\hat{x}_{j,r}(k+1|k)], [\hat{y}_{j,r}(k+1|k)])$ have been defined in (17) and (18), respectively.

This ends the stage of corresponding point identification. The algorithm then advances to the next stage of temporal information filtering.

D. Temporal Information Filtering

To facilitate the tracking scheme, the temporal information between the kth and $(k+1)$th frames is processed. As mentioned earlier, accumulated temporal information up to the $(k+1)$th frame is stored in $\boldsymbol{x}_j(k+1)$ in our work. For convenience, we repeat the trajectory model (6) and the observation model (9) here:

$$\begin{aligned} \boldsymbol{x}_j(k+1) &= \boldsymbol{f}[\boldsymbol{x}_j(k)] + \boldsymbol{w}_j(k+1) \\ \boldsymbol{z}_j(k+1) &= \boldsymbol{H}\boldsymbol{x}_j(k+1) + \boldsymbol{n}_j(k+1). \end{aligned}$$

Then, given the corresponding point $\hat{\boldsymbol{z}}_j(k+1)$, the current stage employs an EKF to obtain the estimate of $\boldsymbol{x}_j(k+1), \hat{\boldsymbol{x}}_j(k+1|k+1)$.

Since $\boldsymbol{f}$ is nonlinear, in order to apply the EKF, the following matrix is first defined:

$$\boldsymbol{F} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}. \tag{36}$$

The temporal information is then processed by performing following steps successively:

- **Step 1**: State and covariance propagation

$$\begin{aligned} \hat{\boldsymbol{x}}_j(k+1|k) =& \boldsymbol{f}[\hat{\boldsymbol{x}}_j(k|k)] \\ \hat{\boldsymbol{P}}_j(k+1|k) =& \boldsymbol{F}[\hat{\boldsymbol{x}}_j(k|k)]\hat{\boldsymbol{P}}_j(k|k)\boldsymbol{F}[\hat{\boldsymbol{x}}_j(k|k)]^T + \boldsymbol{Q}_j(k+1) \end{aligned} \tag{37}$$

  where $\hat{\boldsymbol{x}}_j(k|k)$ is defined in (11). $\hat{\boldsymbol{P}}_j(k|k), \hat{\boldsymbol{P}}_j(k+1|k), \boldsymbol{Q}_j(k+1)$ are covariance matrices of $\hat{\boldsymbol{x}}_j(k|k), \hat{\boldsymbol{x}}_j(k+1|k)$ and $\boldsymbol{w}_j(k+1)$, respectively.

- **Step 2**: State and covariance update

$$\begin{aligned} \boldsymbol{K}_j(k+1) =& \hat{\boldsymbol{P}}_j(k+1|k)\boldsymbol{H}^T \\ & \cdot [\boldsymbol{H}\hat{\boldsymbol{P}}_j(k+1|k)\boldsymbol{H}^T + \boldsymbol{R}_j(k+1)]^{-1} \\ \hat{\boldsymbol{x}}_j(k+1|k+1) =& \hat{\boldsymbol{x}}_j(k+1|k) + \boldsymbol{K}_j(k+1) \\ & \cdot [\hat{\boldsymbol{z}}_j(k+1) - \boldsymbol{H}\hat{\boldsymbol{x}}_j(k+1|k)] \\ \hat{\boldsymbol{P}}_j(k+1|k+1) =& [I - \boldsymbol{K}_j(k+1)\boldsymbol{H}]\hat{\boldsymbol{P}}_j(k+1|k) \end{aligned} \tag{38}$$

  where $\boldsymbol{R}_j(k+1)$ is the covariance matrix of $\boldsymbol{n}_j(k+1)$, and I is the identity matrix. $\boldsymbol{K}_j(k+1)$ represents the gain matrix, while $\hat{\boldsymbol{x}}_j(k+1|k+1)$ is the updated state vector and $\hat{\boldsymbol{P}}_j(k+1|k+1)$ is the associated covariance matrix.

Note that in applying the EKF to process the temporal information, the corresponding point $\hat{\boldsymbol{z}}_j(k+1)$ is assumed to be Normally distributed around $\boldsymbol{H}\hat{\boldsymbol{x}}_j(k+1|k)$. Despite being a heuristic assumption, it works well in our experiments. Moreover, $\hat{\boldsymbol{x}}_j(k+1|k+1)$, in particular the position estimates $(\hat{x}_j(k+1|k+1), \hat{y}_j(k+1|k+1))$, are only used to exploit the temporal information as shown in (13). The observation $\hat{\boldsymbol{z}}_j(k+1)$ should be used as the corresponding point of the jth feature point in the $(k+1)$th frame whenever higher-level applications are considered.

After the temporal information is processed, the task of tracking the jth feature point to the $(k+1)$th frame is completed. The algorithm can now continue tracking the jth feature point to the $(k+2)$th frame. Before presenting the experimental results of the tracking algorithm, we digress a little bit to present a scheme for including new feature points detected in the $(k+1)$th frame in the next section.

IV. Inclusion of New Features

When tracking a set of feature points over a sequence, it is likely that some points disappear after some frames. Accordingly, related tasks may be affected since the amount of information reduces. (Here, each tracked feature point is assumed to provide useful information, although in some applications, only a minimum number of point correspondences is required.) In addition, since the scene in the sequence constantly changes, it is desirable to track feature points in the successive frames which may well contain more information than feature points currently being tracked. We suggest a scheme which considers the problem of including new feature points detected later in the sequence.

Assume that the algorithm has already completed tracking all feature points from the kth frame to the $(k+1)$th frame. Without loss of generality, suppose that there remain M feature points in the $(k+1)$th frame and consequently, there exist M validation gates, $\{\boldsymbol{V}_{1,k+1}(\gamma), \boldsymbol{V}_{2,k+1}(\gamma), \cdots, \boldsymbol{V}_{M,k+1}(\gamma)\}$. Note that the M validation gates are defined similarly according to (14). Then since $\boldsymbol{V}_{j,k+1}(\gamma)$ represents a neighborhood of the corresponding point of the jth feature point, we consider that points inside $\boldsymbol{V}_{j,k+1}(\gamma)$ more or less carry the same information as the jth feature point for other applications. Equivalently, for points extracted from the $(k+1)$th frame by the algorithm reported in [12], only those points outside all of

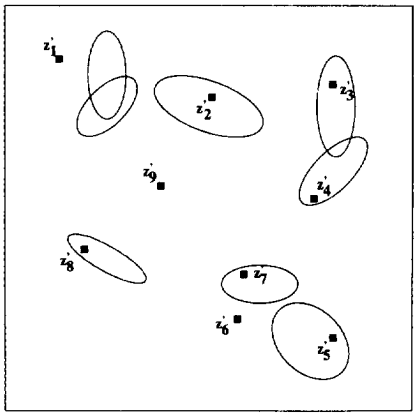

Fig. 4. Illustrations of the scheme for including new feature points: The tracking algorithm tracks eight feature points to the $(k+1)$th frame and form eight validation gates. $z'_1, \cdots, z'_9$ are nine newly extracted feature points. Only z'_1, z'_6 and z'_9 are considered as new feature points of the $(k+1)$th frame.

the M validation gates are regarded as new feature points. The tracking algorithm therefore starts tracking these points only.

For example, consider Fig. 4. Assume that the tracking algorithm maintains eight trajectories in the $(k+1)$th frame. This results in eight validation gates. Meanwhile, the feature extraction algorithm extracts nine feature points from the $(k+1)$th frame. Since only z'_1, z'_6 and z'_9 are outside all the eight validation gates, the algorithm only recognizes these three as new feature points.

As shown in the experiments later, the proposed scheme is quite efficient. It not only handles the problem of the decreasing number of tracked feature points, it also prevents the number of feature points from growing too fast. This is because that when more feature points are tracked, the image region covered by the associated validation gates also enlarges.

This completes the description of the scheme for including new feature points as well as our algorithm for tracking a dynamic set of feature points. Next section presents the experimental results on four real image sequences.

V. Experimental Results

In this section, tracking results are presented for four real image sequences taken by cameras undergoing different types of motion. A tracking list which contains the corresponding points as well as the new feature points is created and updated at every frame. For visual purposes, only the trajectories of the feature points tracked from the first frame as well as the new feature points added to the tracking list at subsequent frames are displayed. The dynamic behavior of the algorithm is shown in a table which lists the number of feature point trajectories being maintained or removed from the tracking list and the number of new points selected from every frame.

A. UMASS PUMA2 Sequence

The first sequence is known as the UMASS PUMA2 sequence; it consists of thirty 256 × 256 frames. The camera is connected to the end of a PUMA robot arm and rotates about a rotation center which is close to the image center. Fig. 5 shows the trajectories for a set of feature points automatically extracted from the first frame by the algorithm reported in [12]; the trajectories are shown up to the 19th and 30th frames. The new feature points extracted by the feature extraction algorithm from frames 3, 19, and 30 are also shown in Fig. 5 in addition to the labeled points which were added to the tracking list at different time instants. The number of feature points being tracked varies with time, as shown in Table I. As seen in Table I, the algorithm for adding new points to the tracking list efficiently maintains the number of points on the list.

TABLE I
Number of Feature Points in the Tracking List for the UMASS PUMA2 Sequence

frame number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
# of points in the list	0	21	19	28	31	35	38	40	41	40	41	43	43	44	46
# of points extracted	23	23	22	26	24	27	29	29	28	28	25	16	19	20	24
# of new points	23	0	9	4	5	5	3	3	4	4	3	1	2	2	5
frame number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
# of points in the list	49	48	49	48	50	49	51	50	53	50	52	52	53	54	53
# of points extracted	18	21	20	21	20	21	20	19	23	25	24	24	27	24	15
# of new points	2	3	1	3	1	3	0	4	0	3	0	3	2	1	1

TABLE II
Number of Feature Points in the Tracking List for the Coke-Can Sequence

frame number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
# of points in the list	0	12	13	16	18	20	22	22	22	26	27	28	28	28	28
# of points extracted	13	12	15	12	13	17	15	15	16	15	14	17	15	17	14
# of new points	13	1	4	2	2	2	1	1	5	1	1	2	0	1	1

TABLE III
Number of Feature Points in the Tracking List for the UMASS Rocket ALV Sequence

frame number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
# of points in the list	0	19	19	23	22	22	19	16	19	21	24	22	22	23	21
# of points extracted	25	20	16	12	14	13	16	16	13	18	14	21	16	17	14
# of new points	25	4	6	1	3	0	7	8	6	6	2	2	3	1	1
frame number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
# of points in the list	19	19	18	18	21	19	24	25	27	29	28	26	27	25	25
# of points extracted	19	19	19	18	22	15	17	16	18	18	17	16	19	19	18
# of new points	2	1	3	5	1	7	1	4	2	3	3	2	4	5	2

B. Coke-Can Sequence

The second sequence is the Coke-Can Sequence, in which the camera is approaching the scene, with the focus of expansion (FOE) located on the coke can. Fifteen frames chosen from the densely sampled sequence (every tenth frame) are used. The original 512 × 512 images are down-sampled to 256 × 256 before applying the algorithm. The resulting trajectories from the first frame to the tenth and 15th frames are shown in Fig. 6. As seen from the figures, because of the pure translation of the camera, the trajectories of the feature points diverge from the FOE. The number of tracked feature points at each time instant is listed in Table II. The new feature points added at the second, tenth, and 15th frames are also marked in Fig. 6.

C. Rocket ALV Sequence

The third sequence is the 30-frame UMASS Rocket ALV Sequence. Again, the 512 × 512 images are down-sampled to 256 × 256 before applying the algorithm. In this sequence, the

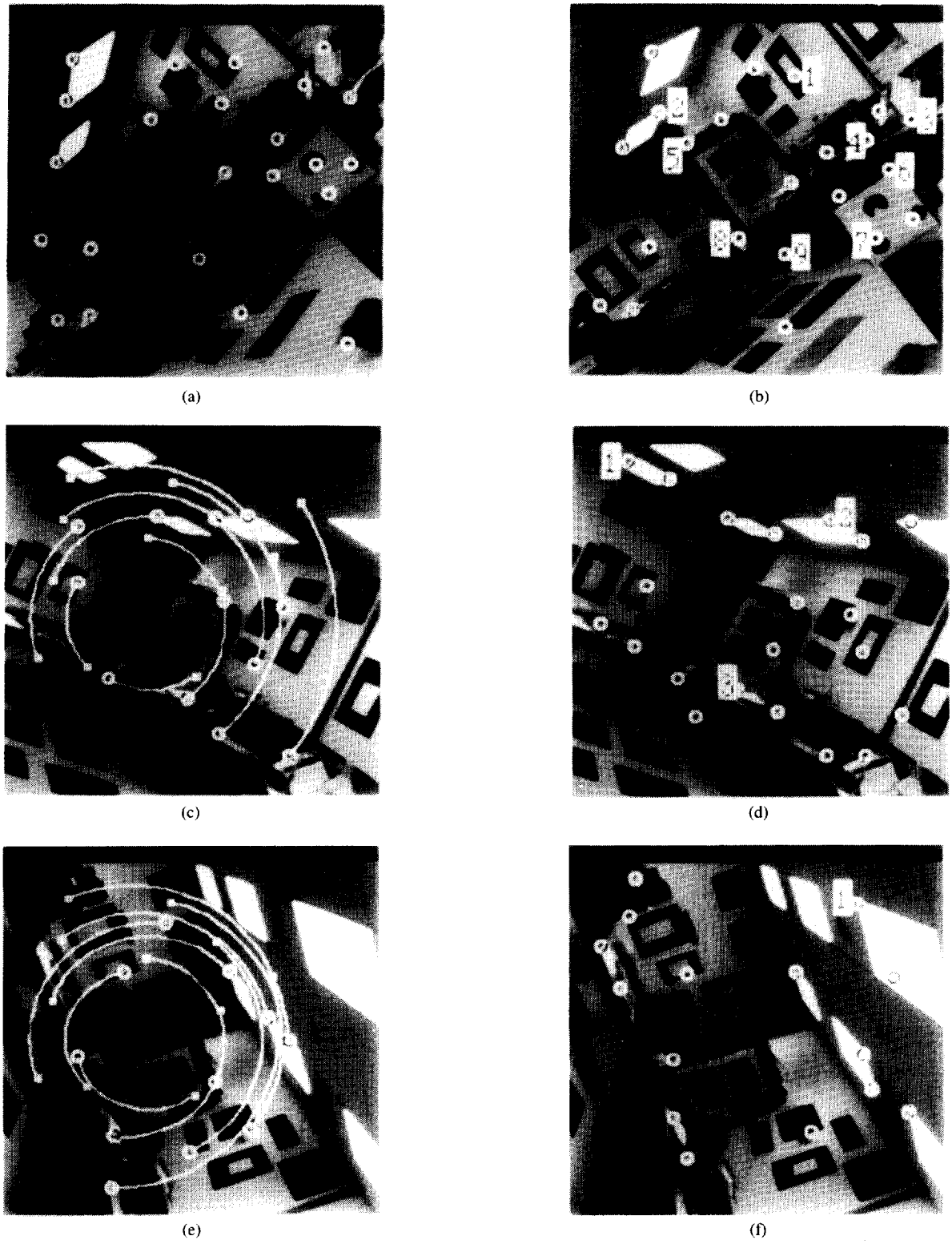

Fig. 5. Tracking results for the UMASS PUMA2 sequence.

camera is mounted on a vehicle which appears to be moving along a straight line to the left and into the image plane with almost no rotation. Due to the uneven terrain, the motion of the camera is not smooth. The trajectories for the feature points up to the 13th and 30th frames are shown in Fig. 7. Table III lists the number of feature points on the tracking list. It is noted that, for the outdoor images acquired from a moving vehicle, the scene close to the camera normally appears in the lower part of the images. A threshold is therefore set to remove the detected points, which are far away such as points on the cloud. The extracted feature points as well as the new points selected by the criterion in Section IV from the second, 13th, and 30th

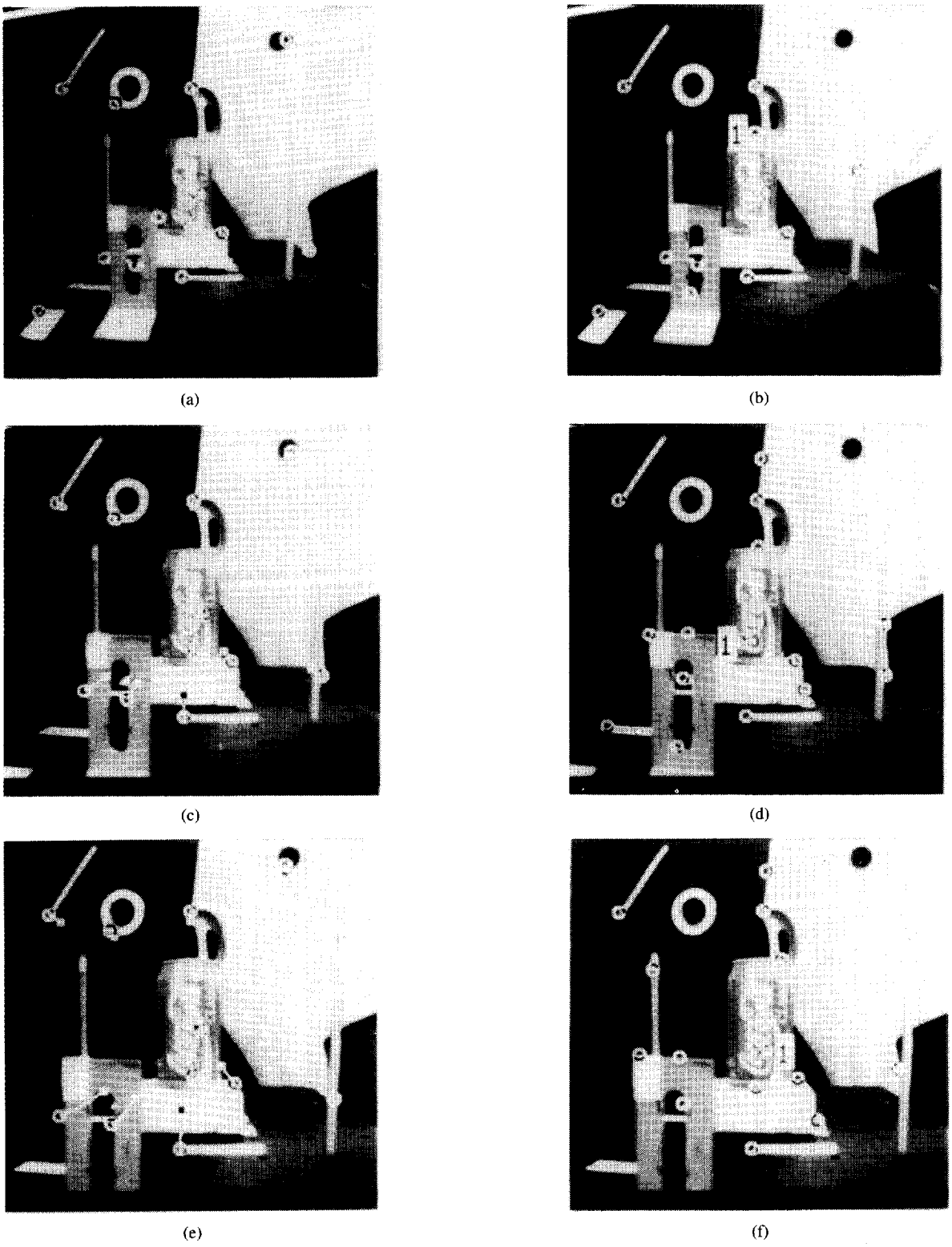

Fig. 6. Tracking results for the Coke-Can Sequence.

frames are also shown in Fig. 7. In this sequence, many feature points move out of the field of view in the first few frames. It is therefore necessary to include new feature points when they become available. In addition, it is apparent from the sequence that the vehicle has an abrupt change in heading direction at the 16th and 20th frames, but the algorithm still keeps tracking most of the feature points.

D. Martin Marietta R3 Sequence

The last sequence is one of the four sequences distributed by Martin Marietta as part of the UGV-RSTA project. As in the third sequence, the camera is mounted on a vehicle and the images are taken when the vehicle is moving through an outdoor environment. The original sequence consists of densely sampled images of size 347×238; only 30 frames

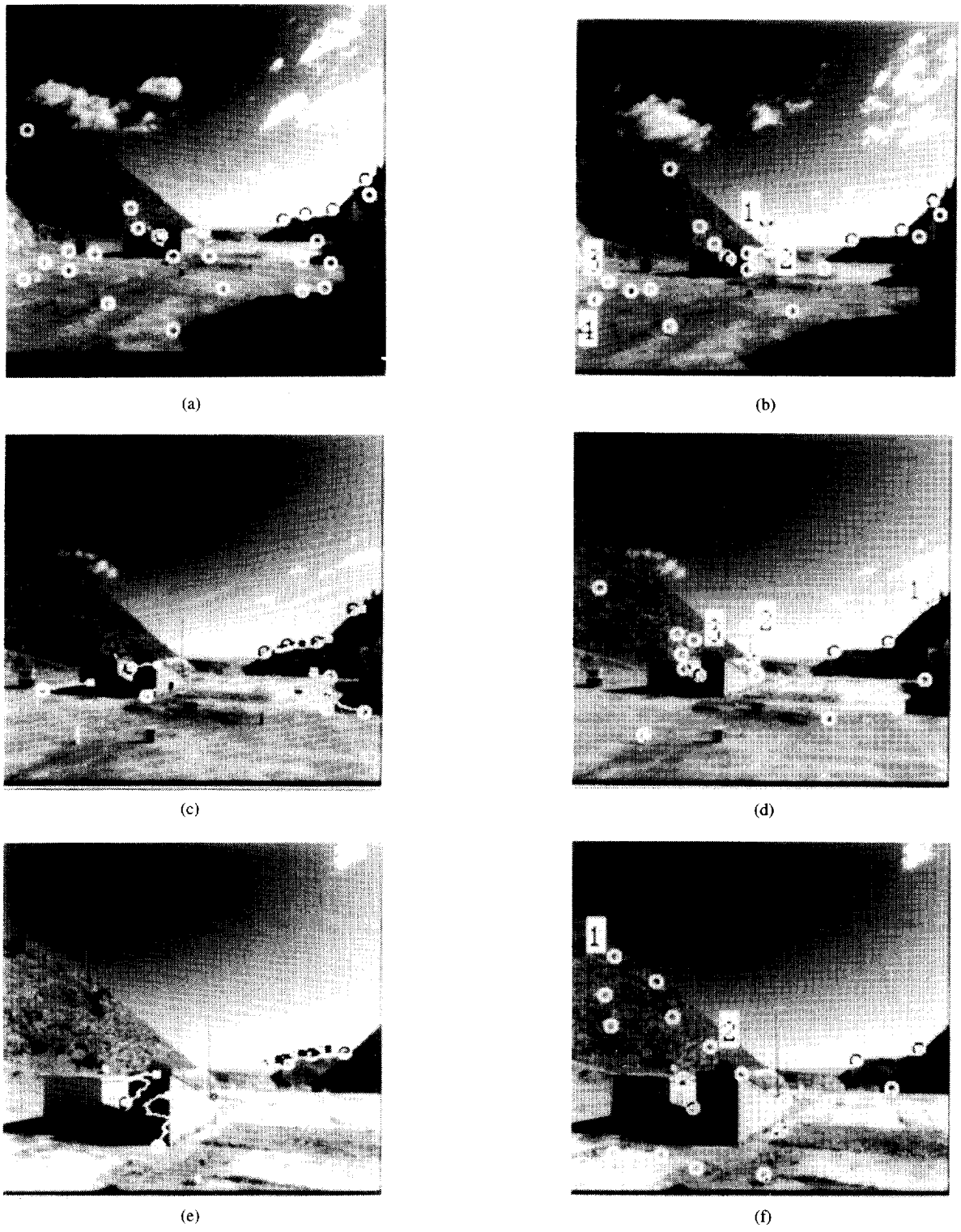

Fig. 7. Tracking results for the Rocket ALV Sequence.

(every fifth frame) in the original sequence were used in the experiment. During the acquisition of the images, the vehicle moves to the right and slightly into the scene. Fig. 8 shows the trajectories of a set of feature points from the first frame to the 19th and 30th frames. As seen from the figures, the points on the mountain are far away from the vehicle, resulting in small movements on the image plane. Fig. 8 also shows the feature points detected in the second, 19th, and 30th frames and the points added to the tracking list. The dynamic inclusion of the new feature points is summarized in Table IV.

VI. Conclusions

An algorithm for tracking a dynamic set of feature points to subpixel accuracy over a sequence of images is presented. To exploit the temporal information contained in a sequence, a simple 2-D kinematic motion model is employed *locally*

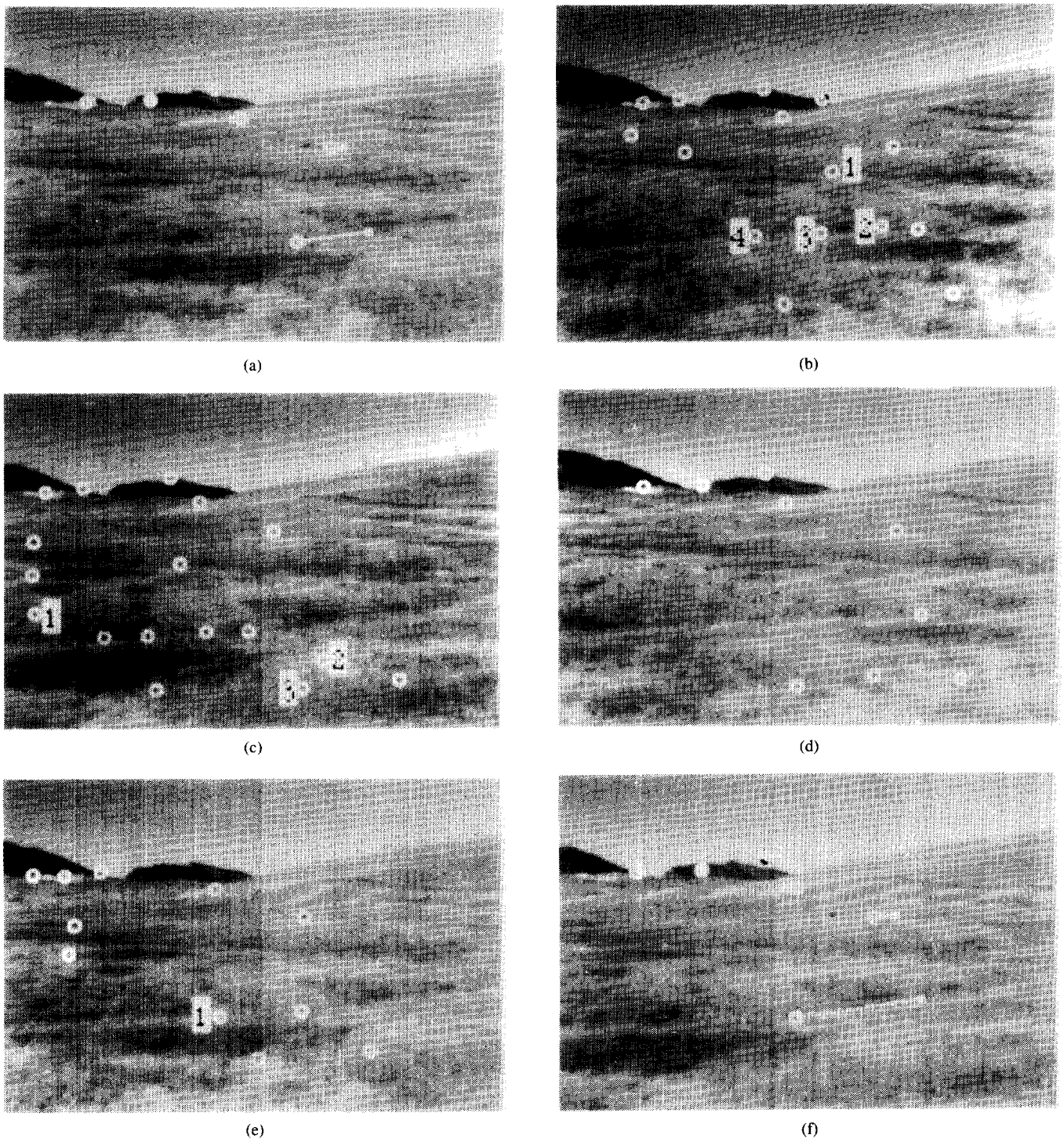

Fig. 8. Tracking results for the Martin Marietta R3 Sequence.

TABLE IV
NUMBER OF FEATURE POINTS IN THE TRACKING LIST FOR THE MARTIN MARIETTA R3 SEQUENCE

frame number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
# of points in the list	0	9	12	18	21	20	23	22	25	25	27	27	30	30	28
# of points extracted	9	15	17	18	15	15	13	14	10	17	11	15	17	10	12
# of new points	9	4	8	3	0	3	0	3	1	4	0	3	2	0	2
frame number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
# of points in the list	27	29	30	31	32	33	33	33	37	38	37	37	37	39	37
# of points extracted	18	19	18	11	15	17	15	17	19	19	20	15	16	10	17
# of new points	2	1	4	1	2	2	1	4	1	2	3	3	5	0	3

to describe the trajectory of each feature point in this work. Due to the consideration of such localized tracking scheme, complicated 3-D ego-motion estimation problems are avoided. On the other hand, to account for nonsmooth changes in the image motion arising from the 3-D movements of the camera, an interframe motion estimation scheme is designed. Accordingly, the algorithm is able to follow the arbitrary movements of feature points. Meanwhile, a scheme which is able to include as well as track new points detected from the subsequent frames is proposed. As shown in the experiments,

the scheme efficiently preserves the information in a sequence which makes the algorithm useful for estimating the pose and ego-motion of the camera.

REFERENCES

[1] J. K. Aggarwal, "Motion and time-varying imagery—An overview," in *Proc. IEEE Workshop Motion: Representation Anal.,* Kiawah Island, SC, May 1986, pp. 1–6.

[2] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association.* San Diego, CA: Academic, 1988.

[3] S. D. Blostein and T. S. Huang, "Detecting small, moving objects in image sequences using sequential hypothesis testing," *IEEE Trans. Signal Processing,* vol. 39, pp. 1611–1629, July 1991.

[4] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Survey,* vol. 24, pp. 325–376, Dec. 1992.

[5] P. Burlina and R. Chellappa, "Time-to-X: Analysis of motion through temporal parameters," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.,* Seattle, WA, June 1994, pp. 461–468.

[6] Y. L. Chang and J. K. Aggarwal, "3D structure reconstruction from an ego motion sequence using statistical estimation and detection theory," in *Proc. IEEE Workshop Visual Motion,* Princeton, NJ, Oct. 1991, pp. 268–273.

[7] I. J. Cox, "A review of statistical data association techniques for motion correspondence," *Int. J. Comput. Vision,* vol. 10, pp. 53–56, Aug. 1993.

[8] N. Cui, J. Weng, and P. Cohen, "Extended structure and motion analysis from monocular image sequences," in *Proc. 3rd Int. Conf. Comput. Vision,* Osaka, Japan, Dec. 1990, pp. 222–229.

[9] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[10] J. Q. Fang and T. S. Huang, "Some experiments on estimating the 3-D motion parameters of a rigid body from two consecutive image frames," *IEEE Trans. Patt. Anal. Machine Intell.,* vol. PAMI-6, pp. 545–554, Sept. 1984.

[11] T. S. Huang, Ed., *Image Sequence Analysis.* Berlin/Heidelberg, Germany: Springer-Verlag, 1981.

[12] B. S. Manjunath, R. Chellappa, and C. V. Malsburg, "A feature-based approach to face recognition," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.,* Champaign, IL, June 1992, pp. 373–378.

[13] I. K. Sethi and R. Jain, "Finding trajectories of feature points in a monocular image sequence," *IEEE Trans. Patt. Anal. Machine Intell.,* vol. PAMI-9, pp. 56–73, Jan. 1987.

[14] J. Weng, N. Ahuja, and T. S. Huang, "Matching two perspective views," *IEEE Trans. Patt. Anal. Machine Intell.,* vol. 14, pp. 806–825, Aug. 1992.

[15] J. Weng, T. S. Huang, and N. Ahuja, "3-D motion estimation, understanding, and prediction from noisy image sequences," *IEEE Trans. Patt. Anal. Machine Intell.,* vol. PAMI-9, pp. 370–389, May 1987.

[16] G. Wolberg, *Digital Image Warping.* Los Alamitos, CA: IEEE Comput. Soc., 1990.

[17] T. H. Wu, R. Chellappa, and Q. Zheng, "Experiments on estimating egomotion and structure parameters using long monocular image sequences," *Int. J. Comput. Vision,* vol. 15, pp. 77–103, June 1995.

[18] Z. Zhang and O. D. Faugeras, "Three-dimensional motion computation and object segmentation in a long sequence of stereo frames," *Int. J. Comput. Vision,* vol. 7, pp. 211–241, Aug. 1992.

[19] Q. Zheng and R. Chellappa, "Automatic feature point extraction and tracking in image sequences for arbitrary camera motion," *Int. J. Comput. Vision,* vol. 15, pp. 31–76, June 1995.

Yi-Sheng Yao received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1989, and the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, in 1991. He is currently working toward the Ph.D. degree in electrical engineering at the University of Maryland, College Park.

He was a Research Assistant at the Signal and Image Processing Institute, University of Southern California, in 1991. Since then, he has been employed by the Computer Vision Laboratory at the University of Maryland as a Research Assistant. His current research interests include signal processing, image processing, and computer vision.

Rama Chellappa (S'78–M'79–SM'83–F'92) is a Professor in the Department of Electrical Engineering at the University of Maryland, where he is also affiliated with the Institute for Advanced Computer Studies, the Center for Automation Research (Associate Director) and Computer Science Department. He has written 20 book chapters and more than 150 peer-reviewed journal and conference papers. Many of his papers have been reprinted in collected works published by IEEE Press, IEEE Computer Society Press, and MIT Press. He is an editor of Collected Papers on Digital Image Processing, (IEEE Computer Society Press, 1992). He is a coauthor of a research monograph on *Artificial Neural Networks for Computer Vision* (Springer Verlag, 1992) and a coeditor of *Markov Random Fields: Theory and Applications* (Academic Press, 1993). His current research interests are in signal and image processing, computer vision, and pattern recognition.

Dr. Chellappa was an Associate Editor for IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING from 1987 to 1989, the IEEE TRANSACTIONS ON NEURAL NETWORKS from 1992 to 1993, and the IEEE TRANSACTIONS ON IMAGE PROCESSING from 1990 to 1993. Currently, he is Co-editor-in-Chief of *Computer Vision, Graphics, and Image Processing: Graphic Models and Image Processing.* He received the 1985 National Science Foundation (NSF) Presidential Young Investigator Award and the 1985 IBM Faculty Development Award. In 1990, he received the Excellence in Teaching Award from the School of Engineering at USC. He is a corecipient of four NASA certificates for his work on synthetic aperture radar image segmentation. He is also a corecipient of the best paper award, presented at the 1992 International Conference on Pattern Recognition. He was the General Chairman of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition and of the IEEE Computer Society Workshop on Artificial Intelligence for Computer Vision, both held in San Diego in June 1989. He was a program cochairman for the NSF-sponsored Workshop on Markov Random Fields, also held in San Diego in June 1989. He was the program chairman of the IEEE Signal Processing Workshop on Neural Networks for Signal Processing held in Baltimore in September 1993 and is the program chairman for the Second International Conference on Image Processing to be held in Crystal City in October 1995.