Proc. Natl. Sci. Counc. ROC(A)
Vol. 23, No. 1, 1999. pp. 151-168

A 3D Feature-Based Tracker for Multiple Object Tracking

CHENG-YUAN TANG*,**, YI-PING HUNG**,†, SHENG-WEN SHIH***, AND ZEN CHEN*

**Institute of Computer Science & Information Engineering*
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.
***Institute of Information Science*
Academia Sinica
Taipei, Taiwan, R.O.C.
****Department of Computer Science & Information Engineering*
National Chi Nan University
Nantou, Taiwan, R.O.C.

(Received March 31, 1998; Accepted May 22, 1998)

ABSTRACT

This paper presents a 3D feature-based tracker for tracking multiple moving objects using a computer-controlled binocular head. Our tracker operates in two phases: an initialization phase and a tracking phase. In the initialization phase, correspondence between 2D features in the first stereo image pair is determined reliably using the epipolar line constraint and *mutually-supported consistency*. In the tracking phase, the feedback loop is established by first predicting new 3D feature locations with Kalman filters (KF) and then projecting them onto the 2D images to guide the extraction of 2D features in the new image pair. Here, we propose a RANSAC (RAN-dom SAmple Consensus)-based clustering method for motion segmentation and estimation using the principle of *rigid body consensus*, which states that all the extracted 3D features on a rigid body should have the same 3D motion. This new method leads to a feature-clustering algorithm which provides a systematic method for managing splitting, merging, appearance and disappearance of multiple moving rigid objects-including articulated objects, such as robot manipulators. Using the motion estimates obtained with the RANSAC-based method as the measurements for the KFs, we are able to use *linear* KFs for predictive visual tracking instead of the commonly-used extended Kalman filters (EKF). Experiments have shown that our tracking system does give good results and can serve as a robust 3D feature tracker for the active binocular vision system we are developing.

Key Words: computer vision, visual tracker, active vision, motion clustering, motion estimation, linear Kalman filters

I. Introduction

Advocated and pioneered by Aloimonos and Bajcsy in the 1980s (Aloimonos *et al.*, 1987; Bajcsy, 1988), active vision is now an important field in computer vision. Just a few years ago, in order to conduct real experiments on different problems in active vision, researchers had to build their own robotic heads, which provided mechanisms for modifying the extrinsic or intrinsic parameters of cameras under computer control (Ahuja and Abbott, 1993; Christensen, 1993; Crowley *et al.*, 1992; Ferrier and Clark, 1993; Fiala *et al.*, 1994; Krotkov and Bajcsy, 1993; Milios *et al.*, 1992; Pahlavan and Eklundh, 1992). Recently, computer controllable binocular heads have become commercially available (e.g., TRC and GEC-Marconi). In the next few years, due to the availability and popularity of experimental equipment, active vision can be expected to receive even more attention from researchers in the fields of computer vision and robotics. There are many interesting and important problems in active vision, including gaze control, attention shift, eye-hand coordination, and object tracking. In this paper, we present a new 3D feature-based predictive visual tracker for tracking multiple moving objects in a cluttered environment using a computer-controlled binocular head.

Our work is closely related to the work of Kitchen

†To whom all correspondence should be addressed.

and Cooper. Cooper (1994) built a 2D feature-based tracker that can track about 35 feature points in an image at a rate of 5 frames per second and organize them into clusters on the basis of their motion. The attributes of the features he used in tracking were not just the location of the feature point, but also an image patch centered at that location. Cheng and Kitchen (1993) extended this 2D tracker to a 3D feature-based tracking system by utilizing information in stereo correspondence. While they used the rigid body constraint in motion clustering and estimation, their motion estimation method was nonlinear, and their motion clustering algorithm was quite primitive, which made their results sensitive to noise and initial guesses. Furthermore, the initial stereo correspondences were selected manually in their system, and the motion model they used could not even describe a constant angular velocity motion, not to say long term motion behavior. Another related work was the tracking system built by Zhang and Faugeras (1992a, 1992b), which used line-based features instead of point features. Although line segments are useful features for tracking, there can be more point features than line features in a natural scene, and it would be better to use all the information available in order to get better results. Also, using line segments only will limit application to classes of images where line features are prevalent (Lew *et al*., 1994). Although our approach can easily be extended to use of both point features and line features, this paper considers only point features.

Recently, the robotics research group of Oxford University developed a high performance head/eye platform (Sharkey *et al.*, 1993). In their system, real-time image processing was made possible by using multiple high speed processors. Based on their binocular head/eye system, they implemented several tracking algorithms for computing optical flow and tracking corner features at very high speed. (The speed ranged from several frames per second to video rate.) Using their optical flow module, they developed an algorithm for controlling saccade and pursuit of an active head/eye platform (Bradshaw *et al*., 1994; Murray *et al*., 1995). Also, based on their corner tracking module, they developed methods for robot navigation and fixation control. Among their works, the one most related to our tracking system described in this paper is the one which tracks foveated feature clusters using an affine structure (Reid and Murray, 1996). In their work, corner features were detected and tracked from frame to frame using a constant image-velocity Kalman filter (KF). The amount of image motion due to ego-motion could be approximately estimated using the controlled motion parameters of the binocular head, which could then be used to eliminate the image motion of the static background. By eliminating the effects of ego-motion, the moving object could be easily segmented, and then a stable fixation point could be determined using an affine structure. This work is interesting because it is fast, reliable, viewpoint invariant, and insensitive to occlusion. However, because they did not consider multiple moving objects, their method will fail when there is more than one moving object. Wavering and Lumia used TRICLOPS (The Real-time Intelligently ControLled, Optical Positioning System), the binocular head they built at the National Institute of Standards and Technology (NIST), to track an object undergoing random or periodic motion (Fiala *et al*., 1994). However, only experiments on one moving object with a simple background were described. Allen *et al*. (1993) used a stationary binocular system to track a single object in real time. Once tracking was stable, the system could command a robot arm to grasp the moving object. Their system relied on real-time stereo-triangulation of optical flow and tried to cope with the inherent noise and inaccuracy of visual sensors by applying a nonlinear filter to recover the correct trajectory parameters. Papanikolopoulos *et al*. used a monocular hand-eye system to track a moving target by introducing adaptive control techniques in order to compensate for inaccurate modeling of the environment, such as depth estimation (Papanikolopoulos, 1992; Papanikolopoulos *et al*., 1992). Their vision system detected motion by computing the optical flow based on the Sum of Squared Differences (SSD) method. Yao and Chellappa (1995) proposed a localized feature tracking algorithm to track a dynamic set of feature points over a long sequence of monocular images. They decomposed the problem of tracking discrete features into several independent and local problems. Another interesting work on monocular visual tracking was the one by Koller *et al*. (1993), where 3D parameterized models were used in 2D predictive matching. Their experiments showed that moving vehicles could be detected and tracked automatically in monocular image sequences from road traffic scenes recorded by a stationary camera. Even though it is possible to perform visual tracking using a monocular image sequence (Koller *et al*., 1993; Papanikolopoulos *et al*., 1992; Yao and Chellappa, 1995; Young and Chellappa, 1990, Zhang and Chellappa, 1995), many researchers who worked on monocular visual tracking before have now shifted to work on binocular visual tracking, e.g., Cheng and Kitchen (1993) and Yang and Leou (1993). The reason is simple. It is easier to "track" (and predict) 3D objects in 3D space than it is in a 2D image domain (Christensen *et al*., 1995; Faugeras, 1993). Using 3D object and motion models, the motion segmentation problem becomes

easier to solve, and occlusion can be easily detected or predicted. Another advantage of using 3D features is that it can simplify 3D object modelling and recognition. Of course, the price one has to pay is, mainly, the stereo correspondence problem, especially the initial stereo correspondence problem.

Stereo correspondence is well-known as a difficult problem. Therefore, in the past, many papers on binocular visual tracking and motion estimation assumed that stereo correspondence and/or 3D feature correspondence over the frames were given (Broida *et al*., 1990; Weng *et al*., 1987), either for every stereo pair (Young and Chellappa, 1990) or at least for the first stereo pair (Cheng and Kitchen, 1993). In order to extract reliable stereo correspondence automatically, some constraints have to be imposed to eliminate ambiguous correspondence. In this paper, in addition to the widely used epipolar line constraint, we also use a *mutually-supported consistency* constraint, which can remove most of the false matches that occur in order to satisfy the epipolar line constraint.

To track the 3D motion of each object, we adopt the KF approach. Use of KF or extended Kalman filters (EKF) (Bar-Shalom and Fortmann, 1988) is quite popular in visual tracking. Broida *et al*. (1990) constructed a state-space model, incorporating both a kinematic and structural state, and formulated a recursive estimation using an iterative EKF which was initialized with the output of a batch algorithm run on the first few frames. Young and Chellappa (1990) described the computer simulation of a tracking system using an EKF that used a number of noisy 3D points assumed to belong to the same rigid object to estimate its motion. Zhang and Faugeras derived some closed-form solutions for some 3D motion models and used them to formulate an EKF to deal with nonlinear measurement equations (Faugeras, 1993; Zhang and Faugeras, 1992a). Azarbayejani and Pentland (1995) reformulated the basic structure-from-motion problem and used an EKF to recover the motion, the pointwise structure, and the focal length from a long image sequence. In this paper, we formulate a *linear* KF for motion tracking using motion estimates as measurements for the KF. The reason why the KF used here can be linear is that we use motion estimates, instead of feature estimates (either 2D or 3D), as the measurement inputs of the KF. That is, we move the nonlinear processing to a module which computes the estimates of 3D motion parameters.

In order to track multiple moving objects with KF, the system has to partition the 3D feature points into several common-motion clusters before applying KF. Here, we encounter a well-known dilemma. That is, if we do not know which feature belongs to which object beforehand, then we can not determine the motion of each object. However, if we do not know the motion of each object, it is hard to decide which feature belongs to which object. In this paper, we propose a RANSAC-based clustering method for solving the 3D motion clustering and estimation problem using the *rigid body consensus* principle (described below). This clustering method is an extension of the RANSAC (RANdom SAmple Consensus) algorithm proposed by Fischler and Bolles (1981). Based on this method, we have successfully developed an autonomous 3D visual tracking system for tracking multiple moving objects with a controlled binocular head. Rigidity implies that the Euclidean distance between any two points on a rigid body will remain unchanged in the next time instant, and the principle of *rigid body consensus* states that all the 3D features on a rigid body should undergo the same 3D motion. Notice that the rigidity property used in this paper is purely from the viewpoint of the observer and is based on only short-time observation. For example, let A, B, and C be three consecutive links of a robot manipulator, where A and B are connected by joint J1, and B and C are connected by joint J2. Suppose that during a short time interval, J2 moved but Jl did not. Then, from the viewpoint of an observer, A & B will form a rigid body together while C is a different rigid body undergoing a different motion. Ullman *et al*. used the constraint of rigid body to align an object model to an image (or to some smooth surfaces) (Basri and Ullman, 1988; Huttenlocher and Ullman, 1990; Shoham and Ullman, 1988; Ullman and Basri, 1991). However, they did not apply the rigid body principle to motion segmentation. Furthermore, the RANSAC technique has been used in many different applications (Clarke and Zisserman, 1996; Fischler and Bolles, 1981; Gee and Cipolla, 1996; Meer *et al*., 1991; Roth and Levine, 1993; Zhang *et al*., 1995), but to our knowledge, it has never been used to solve the problem of motion segmentation.

Roughly speaking, our tracker consists of the following five components: feature extraction, 2D temporal correspondence, stereo correspondence, motion clustering and estimation, and 3D feature prediction by means of KF. We have organized this paper as follows. Section II presents the system overview and the main algorithm. Section III describes the temporal and stereo correspondence algorithms used in our tracker. In Section IV, we propose an integrated method for motion clustering and estimation based on the RANSAC algorithm. Section V describes the motion kinematics and the formulation of the KF used in our tracker. Experimental results are given in Section VI, and conclusions in Section VII.

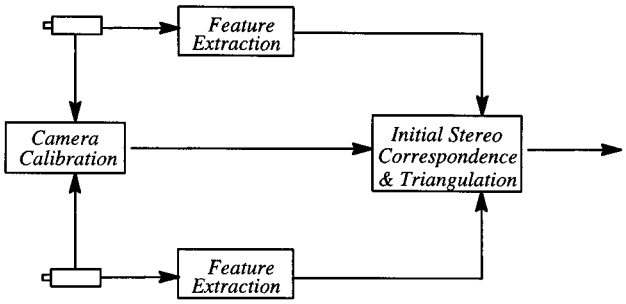


Fig. 1. The system block diagram of the initialization phase.

II. System Overview

The 3D object tracking system operates in two phases: an initialization phase and a tracking phase. Figure 1 shows the system block diagram of the initialization phase, and Fig. 2 shows the system block diagram of the tracking phase. Initially, the stereo cameras are calibrated, in an automatic way, to obtain a 3D measurement accuracy of 1 millimeter (assuming good feature detection) (Shih *et al.*, 1998a, 1998b). The calibration parameters of the stereo cameras are used in the subsequent binocular visual tracking. In the initialization phase, salient features are extracted from the first stereo image pair, and initial stereo correspondences are determined using both the epipolar line constraint and the mutually-supported consistency constraint.

In the tracking phase, the 2D temporal matching module uses the prediction from the previous images to locate possible 2D feature positions in the new images. Based on the possible 2D corresponding features in the stereo image pair, the stereo correspondence module uses the epipolar line constraint and the mutually-supported consistency constraint to reduce the number of ambiguous and error-prone matches. Once stereo correspondences are obtained, we can compute, by stereo triangulation, the corresponding 3D feature positions at the current time instant. With the set of all 3D feature correspondences between two successive image pairs, we then apply the *rigid body consensus* principle to solve the motion clustering and estimation problems simultaneously. Using the estimates of the successive motions obtained from the motion clustering/estimation module as the measurements for the KF, we can predict the 3D positions of the features at the next time instant, which are then projected onto 2D image planes to provide information required by the 2D temporal matching module. The 2D temporal matching module uses this information to determine the search region for the best match in order to reduce the amount of searching time needed. The main algorithm for our 3D feature-based tracker is described below.

Main Algorithm

Step 1: At time t=0 {the initialization phase}
For the first image pair
Step 1.1: Extract 2D features using **Algorithm 1** listed in Appendix 1 with a larger threshold to allow only salient features.
Step 1.2: Solve the initial stereo correspondence problem using **Algorithm 2** (see Section III.1 and in Appendix 2), and compute the initial estimates of the 3D feature locations using stereo triangulation.
Step 2: At time t=1, 2, ...{the tracking phase}
For each new image pair
Step 2.1: Extract 2D features using **Algorithm 1** with a lower threshold (i.e., to allow more candidates in).
Step 2.2: **IF** t=1, **THEN**
For each 3D feature in track, set its 3D prediction $\tilde{p}(1)$ at time t=1 to be the 3D estimate $\hat{p}(0)$ ob-

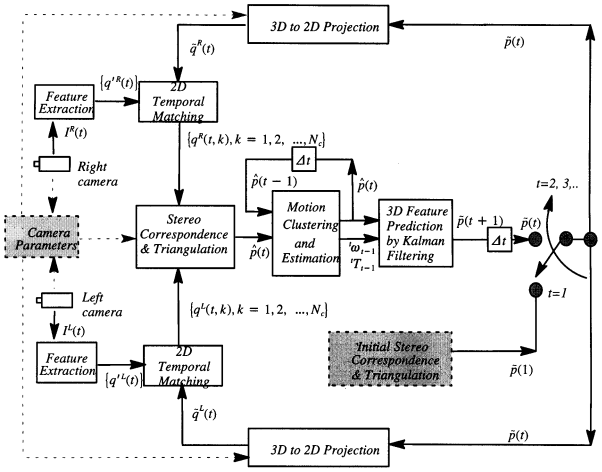


Fig. 2. The system block diagram of the tracking phase (t=1, 2, ...). The notations used are described below: $\hat{p}(t)$ is the estimate of the 3D coordinates of a 3D feature point P in track at time t. ${}^{t}\omega_{t-1}$ and ${}^{t}T_{t-1}$ are the estimates of the rotational and translational motion of the rigid body to which P belongs. $\tilde{p}(t+1)$ is the 3D prediction of P at time t+1, obtained by using the motion estimates from the Kalman filter. $\tilde{q}^{L}(t)$ and $\tilde{q}^{R}(t)$ are the 2D projections of the 3D predicted feature $\tilde{p}(t)$ in the left and right images, respectively. $\{q'^{L}(t)\}$ and $\{q'^{R}(t)\}$ are the sets of 2D features extracted with a lower threshold. $\{q^{L}(t,k), k=1, 2, ..., N_c\}$ and $\{q^{R}(t,k), k=1, 2, ..., N_c\}$ are the candidate sets of temporal correspondence in the left and right images, respectively, where N_c is the number of the candidates. $\hat{p}(t)$ is the estimate of the coordinates of the 3D feature point P at time t.

tained in **Step 1.2** (i.e., assuming zero initial object velocity).
ELSE (t=2, 3, ...)
For each 3D feature in track, set its 3D prediction $\tilde{p}(t)$ to be the one estimated in **Step 2.7** at time $t-1$.

Step 2.3: For each 3D feature in track, compute the 2D projections in both the left and right images.

Step 2.4: For both the left and right images: For each 3D feature in track, use the 2D temporal matching module described in **Algorithm 4** (see Section III.2 and Appendix 3) to generate a set of candidates of temporal correspondence.

Step 2.5: For each 3D feature in track, solve the stereo correspondence problem and compute its 3D estimated coordinates from the outputs of the left and right 2D temporal matching modules using **Algorithm 3** (see Section III.3 and Appendix 2).

Step 2.6: Use the RANSAC-based clustering method to segment multiple moving objects and to estimate their 3D motion simultaneously:
IF t=1, {The initial clustering stage}
THEN
Use **Algorithm 5** (see Section IV and Appendix 4) to partition the 3D features into different common-motion clusters and, possibly, one un-clustered set.
ELSE (t=2, 3, ...) {The cluster maintenance stage}
Maintain clusters of 3D features using **Algorithm 8** (see Section IV and Appendix 4).

Step 2.7: For each common-motion cluster, use a Kalman filter to predict its next movement (see Section V), which is then used to predict the 3D feature locations, $\{\tilde{p}(t+1)\}$, in the next time instant.

END of Main Algorithm

III. Temporal and Stereo Correspondence

A slightly modified version of the corner detector developed by Cooper *et al*. (1993) has been used to extract salient features. Given an intensity image, this feature extraction algorithm (details are given in Appendix 1) can extract a set of corner features. This algorithm may obtain multiple responses in the neighborhood of corner points; therefore, a form of "non-maximum suppression" is used to eliminate the redundant feature points. With this algorithm, a corner feature is accepted as a *suppressor feature* if it is the strongest corner within its suppression reign. Otherwise, it is suppressed by suppressor features and will be referred to as a *suppressed feature*. Notice that both suppressor features and suppressed features are *salient features* and will be used in the stereo correspondence module, but in a different way (see Section III.1). For each feature point, the graylevel values in a small neighborhood are stored as a template for matching.

1. Initial Stereo Correspondence

Let Q^L and Q^R be the two sets of feature points extracted from the left and right images, respectively, at time t=0. If $q^L \in Q^L$ and $q^R \in Q^R$ are the 2D projections of the same 3D feature, then the two small image patches centered at q^L and q^R should look the same, provided that the object surfaces are Lambertian, and that there is no occlusion. Therefore, we can determine if the two feature points, q^L and q^R, are a stereo pair by checking whether the image patches centered at q^L and q^R are similar. A popular similarity measure used to find stereo correspondence is the sum of absolute difference (SAD):

$$\boldsymbol{J} = \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}\left|\boldsymbol{I}^L(i,j) - \boldsymbol{I}^R(i,j)\right|, \qquad \textbf{(1)}$$

where $\boldsymbol{I}^L(i,j)$ and $\boldsymbol{I}^R(i,j)$ are the image patches centered at q^L and q^R, respectively, and $M \times M$ is the size of the image patches.

Because the stereo cameras are well-calibrated, we can use the epipolar-line constraint to restrict the search region when finding stereo correspondence using template matching. However, there may still be ambiguous stereo correspondences if only the epipolar-line constraint is used. In this paper, we propose a new constraint of *mutually-supported consistency* to eliminate unreliable stereo correspondences. To explain this constraint, let us consider Fig. 3. For a suppressor feature A in the left image, we can find the best match B in the right image using template matching. Here, B can be either a suppressor feature or a suppressed feature. Such a stereo correspondence initiated by a suppressor feature in the left image is referred to as a right-directed stereo correspondence pair and will be

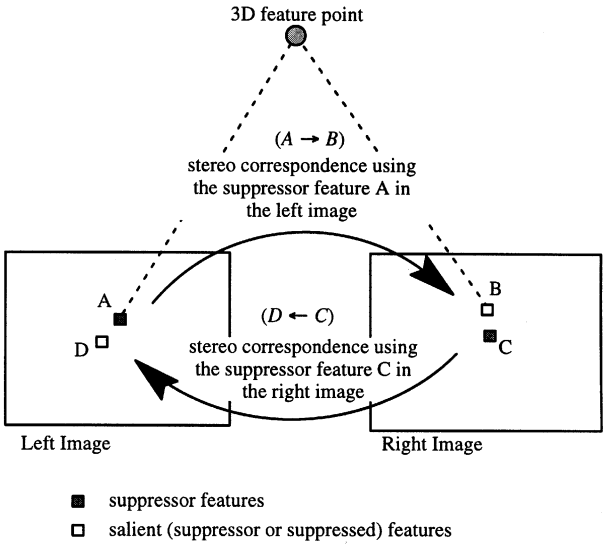


Fig. 3. Illustration of the mutually-supported consistency constraint.

denoted by ($A \rightarrow B$). On the other hand, a left-directed stereo correspondence pair, ($D \leftarrow C$), is a stereo correspondence initiated by a suppressor feature C in the right image, where D is the best match (either a suppressor or suppressed feature) in the left image using template matching. A right-directed stereo correspondence pair ($A \rightarrow B$) is regarded as a *mutually-supported stereo pair* if there exists a left-directed stereo correspondence pair ($D \leftarrow C$) such that both the distance between B and C and the distance between A and D are smaller than a tolerance value. In our tracker, only mutually-supported stereo correspondence pairs are used to compute 3D features using triangulation. The details of the **Initial Stereo Correspondence** algorithm are given in Appendix 2.

2. Temporal Correspondence

Consider Fig. 2. Let P be a 3D feature point in track. Before taking the new stereo image pair, $I^L(t)$ and $I^R(t)$, we have a prediction of the 3D coordinates of P at time t, denoted by $\tilde{p}(t)$. With the known camera parameters and $\tilde{p}(t)$, we can predict the image location of the 3D point P in the left and right images, $I^L(t)$ and $I^R(t)$, respectively. Let the 2D predictions of $\tilde{p}(t)$ in the left and right images be denoted by $\tilde{q}^L(t)$ and $\tilde{q}^R(t)$, respectively. The 2D temporal matching module shown in Fig. 2 is used to find the temporal correspondence. That is, for each $\tilde{q}^L(t)$ (or $\tilde{q}^R(t)$), the 2D temporal matching module finds possible temporal correspondences $\{q^L(t,k), k=1, 2, ..., N_c\}$ (or $\{q^R(t,k), k=1, 2, ..., N_c\}$) using template matching. Notice that N_c depends on the Signal-Noise Ratio (SNR). If the SNR is small, which means the signal is corrupted significantly by noise, then more candidates should be selected to have higher probability of finding a correct match. In this paper, N_C is selected to be 3. The **Temporal Correspondence** algorithm for the 2D temporal matching is described in Appendix 3. To reduce the computation cost, the size of the search region used in the 2D temporal matching module is adaptive to the data. That is, it should depend on the uncertainty of 3D prediction, $\tilde{p}(t)$, which is in turn a function of the uncertainty of the motion estimation.

3. Stereo Correspondence

At the output stage of the 2D temporal matching module, each 3D feature in track may have multiple temporal matching candidates in the left and right images. We use the stereo correspondence module to resolve the unique stereo pair from the multiple temporal candidates in both images and then compute the estimate of the 3D feature location using stereo triangulation. As in the initial stereo correspondence, both the epipolar line constraint and the mutually-supported consistency constraint are utilized here. The details of the **Stereo Correspondence and Triangulation** algorithm are given in Appendix 2.

IV. Clustering of Moving Objects

To track multiple rigid objects, our system partitions 3D feature points into several clusters, each having a common 3D motion. Once clustering is done, for each cluster of 3D feature points, a motion model can be used to predict the locations of the 3D features at the next time instant. Due to modeling error and measurement noise, the 3D prediction may not be good enough, and the temporal and stereo correspondences for some 3D features may occasionally be wrong and may jeopardize the performance of the tracking system. In this section, we will propose a new method for verifying 3D feature correspondences over time and for clustering 3D moving features, based on the principle of rigid body consensus.

Suppose there are N 3D features in track at time $t-1$, i.e., before taking the stereo image pair of $I^L(t)$ and $I^R(t)$. Let $\hat{p}_i(t-1)$, $i=1, 2, ..., N$, be the estimates of the 3D feature locations at time $t-1$. After taking the new stereo image pair $I^L(t)$ and $I^R(t)$ and performing **Step 2.1-Step 2.5** of the **Main Algorithm** described in section II, the system will obtain a set of new estimates for the 3D feature locations at time t, i.e., $\hat{p}_i(t)$, $i=1, 2, ..., N$. Let the set of 3D correspondences between time $t-1$ and time t be denoted by

$$S(t-1,t)=\{(\hat{p}_i(t-1),\hat{p}_i(t))|i=1,\ 2,\ ...,\ N\}. \quad \textbf{(2)}$$

In the following, we can use S to represent $S(t-1,t)$ for simplicity when doing so causes no confusion. Our goal is to partition the 3D feature points into common-motion clusters by using the information contained in $S(t-1,t)$, $t=1,\ 2,\ 3,\ ....$ Simple clustering methods, such as k-means clustering, do not work due to the involvement of rotation motion. Here, we propose a RANSAC-based clustering method which can solve the motion clustering and estimation problems, simultaneously and robustly, by using the principle of *rigid body consensus* described in Section I.

A 3D rigid body motion can be uniquely characterized by a rotation matrix R and a 3D translation vector T. For any three non-collinear 3D feature points, their 3D correspondence over time can be used to compute a least square solution to the motion parameters, R and T, using the Arun method (Arun *et al*., 1987). The strategy we use to cluster 3D feature points in the initial clustering stage ($t=1$) is different from that used in the cluster maintenance stage ($t=2,\ 3,\ ...$). In the initial clustering stage, 3D feature points are partitioned into common-motion clusters by **Algorithm 5**, whose details are given in Appendix 4. In the cluster maintenance stage, points having inconsistent 3D motion will be removed from the existing clusters and merged into an un-clustered set S^0. If the number of un-clustered points in S^0 exceeds $N_{\min}$, *the minimum size required for a consensus set*, then the system will try to form new common-motion clusters from the un-clustered set S^0, which indicates the appearance of new moving objects (or clusters). Splitting and merging of clusters are also taken care of by our cluster maintenance algorithm.

After executing the initial clustering algorithm, each 3D feature point will be either a *member* or a *candidate* of a common-motion cluster, or will still remain an un-clustered feature. Suppose that at time $t-1$, we have $L(t-1)$ common-motion clusters. Let $S_M^j(t-1,t)$ be the set of all 3D correspondences $(\hat{p}_i(t-1),\ \hat{p}_i(t))$ such that their corresponding 3D feature points are *members* of the jth common-motion cluster. Let $S_C^j(t-1,t)$ be the set of all 3D correspondences $(\hat{p}_i(t-1),\ \hat{p}_i(t))$ such that their corresponding 3D feature points are *candidates* of the jth common-motion cluster. Let

$$S^j(t-1,t)\equiv S_M^j(t-1,t)\cup S_C^j(t-1,t).$$

Then,

$$S(t-1,t)=S^0(t-1,t)\cup S^1(t-1,t)\cup ...\cup S^{L(t-1)}(t-1,t),$$

where $S^0(t-1,t)$ is the set of all un-clustered 3D correspondences. Each common-motion cluster contains the following data: a unique cluster identification denoted by j, a common 3D motion denoted by ${}^tR_{t-1}^j$ and ${}^tT_{t-1}^j$, and a set of 3D correspondences over time denoted by $S^j(t-1,t)$, which can be divided into a member set $S_M^j(t-1,t)$ and a candidate set $S_C^j(t-1,t)$. When a new image pair is acquired, the new 3D correspondences over time can be obtained by using the algorithms described in Section III, and then the common-motion clusters have to be updated. Re-clustering of the new 3D correspondences over time is used to manage the splitting and merging of common-motion clusters. For a 3D feature P_i with 3D correspondence $(\hat{p}_i(t-1),\ \hat{p}_i(t))$, we define its *motion similarity error* with respect to cluster j having motion ${}^tR_{t-1}^j$ and ${}^tT_{t-1}^j$ as

$$e_i^j \equiv \left\| \hat{p}_i(t) - {}^tR_{t-1}^j \hat{p}_i(t-1) - {}^tT_{t-1}^j \right\|. \quad \textbf{(3)}$$

If the motion similarity error of a 3D feature P_i with respect to cluster j, e_i^j, is the smallest among all the active clusters, then this 3D feature P_i is said to be *most similar in motion* to cluster j, and cluster j is the *most similar-in-motion cluster* for P_i. For each un-clustered 3D feature, if its motion similarity error with respect to the most similar-in-motion cluster is less than *Tolerance_Loose*, then it is assigned as a candidate of that most similar-in-motion cluster. For each candidate in the jth common-in-motion cluster, if its motion similarity error with respect to the jth cluster is smaller than *Tolerance_Tight*, then it is promoted to be a member. Also, two common-motion clusters, j and k, are said to have common 3D motion if the following two conditions are both satisfied:

$$\frac{1}{\#(S_M{}^j)} \sum_{(\hat{p}(t-1),\hat{p}(t))\in S_M{}^j} \left\| \hat{p}(t) - {}^tR_{t-1}^k \hat{p}(t-1) - {}^tT_{t-1}^k \right\|$$

$$< Tolerance\_Tight, \quad \textbf{(4)}$$

$$\frac{1}{\#(S_M{}^k)} \sum_{(\hat{p}(t-1),\hat{p}(t))\in S_M{}^k} \left\| \hat{p}(t) - {}^tR_{t-1}^j \hat{p}(t-1) - {}^tT_{t-1}^j \right\|$$

$$< Tolerance\_Tight. \quad \textbf{(5)}$$

If two common-motion clusters have common 3D motion for three consecutive cycles, then they are merged. The details of the algorithm for maintaining the common motion clusters are given in **Algorithm 8** (see Appendix 4).

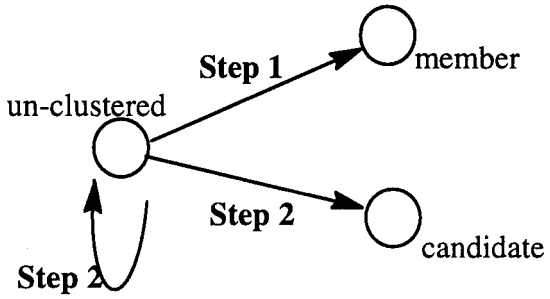


Fig. 4. State transition of a 3D feature in the initial clustering stage (**Algorithm 5**).

Notice that in **Algorithm 8**, **Step 1.2** and **Step 4.2** deal with the disappearance of an active cluster; **Step 3** results in the appearance of new clusters; **Step 4.1** deals with the merging of clusters; and **Step 1**, together with **Step 3**, accomplishes splitting of an active cluster. In our algorithm, only members contribute to motion estimation of a common-motion cluster; candidates are assigned to a common-motion cluster only for prediction of their next 3D position and due to the possibility that they will become members of that cluster in the next time instant. Consider Figs. 4 and 5. Initially, all the 3D feature points are un-clustered. In the initial clustering stage (i.e., **Algorithm 5**), each 3D feature point is classified either as a member of a cluster (**Step 1**) or as a candidate of a cluster (**Step 2**), or it is left un-clustered (**Step 2**). In the cluster maintenance stage (i.e., **Algorithm 8**), a member can either remain in the same cluster (**Step 1.1**) or be merged into the un-clustered set (**Step 1.2** or **Step 4.2**); a candidate can either be promoted as a member (**Step 1.2.2**) or merged into the un-clustered set (**Step 1.2.2** or **Step 4.2**); and an un-clustered feature can either be assigned as a member of a new cluster (**Step 3**) or assigned as a candidate of a cluster (**Step 2**), or it can remain as an un-clustered feature (**Step 2**). Before becoming a member of a common-motion cluster, a 3D feature has to first become a candidate of that cluster unless it is one of the initiators of that cluster. In one iteration of the clustering maintenance algorithm, a member can become a candidate only by means of both **Step 1.2** and **Step 2**; i.e., it can not be demoted to be a candidate directly in one step. This leaves the possibility of letting this member become a candidate of a cluster which is more similar-in-motion than the present one. Similarly, a candidate of a common-motion cluster can remain a candidate only by means of **Step 1.2.2** and **Step 2**; i.e., it has to first be designated as un-clustered in **Step 1.2.2**; then, it can become a candidate again in **Step 2**. This also gives it an opportunity to become a candidate of a more similar-in-motion cluster.

V. Motion Prediction by Kalman Filters

The motion clustering and estimation module described in Section IV will give a new motion estimate for each common-motion cluster based on the new observation of 3D features. The motion estimate can then be used as the new measurement for the KF, which will be used to predict the next motion of the common-motion cluster, as described in this section. The predicted motion will then be used to predict the 3D feature locations in the next time instant. Using these 3D feature predictions, 2D feature matching can be greatly simplified, and much better tracking performance can be achieved. To describe the KF for motion prediction, we need to address some of the modeling issues related to motion kinematics.

1. The Motion Kinematic Model

Let P be a point on a rigid body. Let $p(t)$ and $p(t-1)$ be the 3D coordinates of P at time t and time $t-1$ with respect to the Viewer Reference Frame (VRF). Then, we have the following equation:

$$p(t)={}^{t}R_{t-1}p(t-1)+{}^{t}T_{t-1}, \quad \textbf{(6)}$$

where ${}^{t}R_{t-1}$ is a 3×3 orthogonal matrix specifying the 3D rotation from time $t-1$ to time t, and ${}^{t}T_{t-1}$ is a 3×1 vector specifying the 3D translation from time $t-1$ to time t.

As described by Zhang and Faugeras (1992a) in Chapter 14, a common approach to modeling motion kinematics is to divide the 3D motion of a rigid body into two parts: a rotation around a point (called the center of rotation) and a translation of the center of rotation. Let Q_0 be a fixed 3D point on the rotation

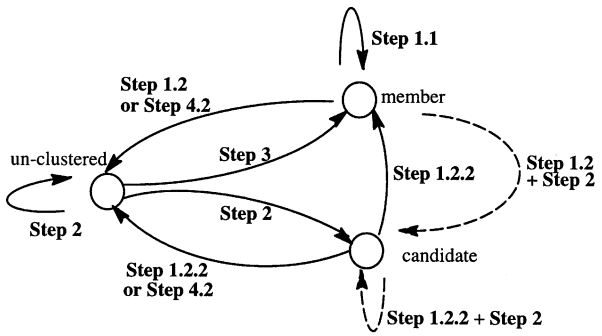


Fig. 5. State transition of a 3D feature in one iteration of the clustering maintenance stage (**Algorithm 8**).

axis, which will be referred to as the *rotation center*. Let $b(t)$ and $b(t-1)$ be the 3D coordinates of Q_0 at time t and time $t-1$ with respect to the VRF; then,

$$b(t)={}^{t}R_{t-1}b(t-1)+{}^{t}T_{t-1}. \quad \textbf{(7)}$$

From Eqs. (6) and (7), we have

$$p(t)-b(t)={}^{t}R_{t-1}[p(t-1)-b(t-1)]. \quad \textbf{(8)}$$

The trajectory of the rotation center, $b(t)$, can be described by the following recursive equation:

$$b(t)=b(t-1)+v(t-1)\Delta t+\frac{1}{2}a(t-1)(\Delta t)^2, \quad \textbf{(9)}$$

where $v(t-1)$ and $a(t-1)$ are the translational velocity and acceleration, respectively, and $v(t)=v(t-1)+a(t-1)\Delta t$. The angular velocity of a 3D object at time t can be denoted by a 3×1 rotation vector $\omega(t)$, whose direction is that of the rotation axis and whose norm is equal to the rotation angle. Then,

$$\omega(t)=\omega(t-1)+\mu(t-1)\Delta t, \quad \textbf{(10)}$$

where $\omega(t-1)$ and $\mu(t-1)$ are the angular velocity and acceleration, respectively. In this paper, we assume constant acceleration, i.e., $a(t)=a(t-1)$ and $\mu(t)=\mu(t-1)$.

2. 3D Feature Prediction by Using KF

In our system, KFs are used to predict 3D motions of multiple rigid objects. The predicted 3D motions are then used to predict the 3D feature locations for the purpose of simplifying 2D temporal matching (or "tracking") in the next time instant. Linear KFs, instead of EKFs, can be applied by formulating the problem in the following way.

The state vector s_t at time t is a 15×1 vector defined as

$$s_t\equiv[\omega(t)^T,\ \mu(t)^T,\ b(t)^T,\ v(t)^T,\ a(t)^T]^T, \quad \textbf{(11)}$$

where $\omega(t)$, $\mu(t)$, $b(t)$, $v(t)$ and $a(t)$ are 3×1 vectors representing the angular velocity, the angular acceleration, the position of the rotation center, the translational velocity and the translational acceleration, respectively, at time t.

Using the constant acceleration assumption and Eqs. (9) and (10), the state equation can be written as

$$s_{t+1}=H\ s_t+n_t, \quad \textbf{(12)}$$

where the state transition matrix H is

$$H=\begin{bmatrix} I_3 & (\Delta t)I_3 & 0 & 0 & 0 \\ 0 & I_3 & 0 & 0 & 0 \\ 0 & 0 & I_3 & (\Delta t)I_3 & \frac{(\Delta t)^2}{2}I_3 \\ 0 & 0 & 0 & I_3 & (\Delta t)I_3 \\ 0 & 0 & 0 & 0 & I_3 \end{bmatrix}, \quad \textbf{(13)}$$

and the random disturbance n_t is white with covariance matrix Q_t, i.e.,

$$E[n_t]=0 \text{ and } E[n_t n_t^T]=Q_t. \quad \textbf{(14)}$$

An important step in applying *linear* KF to the motion prediction problem is to use the motion estimates obtained by the motion clustering and estimation module instead of directly using information from observation of the 3D features. Our goal is then to establish a linear relation between the motion estimates (i.e., measurements) and the system state, s_t. Notice that the motion of the rotation center, $b(t)$, is independent of the angular velocity $\omega(t)$ of the 3D object. From Eqs. (7) and (9), we have

$${}^{t}T_{t-1}=(I_3-{}^{t}R_{t-1})b(t)+{}^{t}R_{t-1}v(t)\Delta t-\frac{1}{2}{}^{t}R_{t-1}a(t)(\Delta t)^2 \quad \textbf{(15)}$$

using $v(t)=v(t-1)+a(t-1)\Delta t$ and $a(t)=a(t-1)$. Also, we can directly compute the angular velocity ${}^{t}\omega_{t-1}$ from ${}^{t}R_{t-1}$ using the Rodrigues formula (Kanatani, 1990). Define the measurement vector as

$$x(t)\equiv\begin{bmatrix} {}^{t}\omega_{t-1} \\ {}^{t}T_{t-1} \end{bmatrix}. \quad \textbf{(16)}$$

Using Eq. (15), the measurement equation can be written as

$$x(t)=F_t s_t+\eta_t, \quad \textbf{(17)}$$

where

$$F_t=\begin{bmatrix} I_3 & (\Delta t)I_3 & 0 & 0 & 0 \\ 0 & 0 & I_3-{}^{t}R_{t-1} & (\Delta t){}^{t}R_{t-1} & -\frac{1}{2}(\Delta t)^2\,{}^{t}R_{t-1} \end{bmatrix}, \quad \textbf{(18)}$$

and the measurement noise η_t is white with covariance

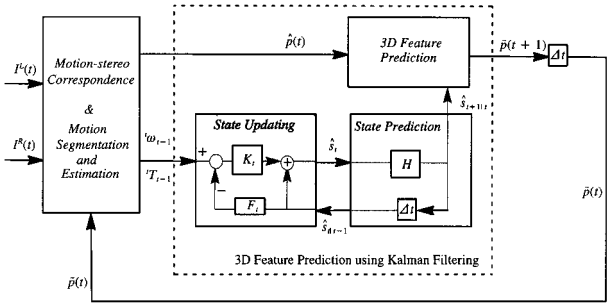


Fig. 6. The block diagram of the module for 3D Feature Prediction using KF and its interface with the rest of the system shown in Fig. 2.

matrix Λ_t, i.e.,

$$E[\eta_t]=0 \text{ and } E[\eta_t \eta_t^T]=\Lambda_t. \quad \textbf{(19)}$$

Figure 6 shows the block diagram of the module for 3D feature prediction using KF and its interface with the rest of the system (also refer to Fig. 2). At time t, based on the new stereo images (i.e., $I^L(t)$ and $I^R(t)$) and the 3D feature predictions from the previous time instant (i.e., $\{\tilde{p}(t)\}$), the rest of the system can generate a set of 3D feature observations at time t,$\{\hat{p}(t)\}$, and then update their clustering (motion segmentation) and estimate the motion, ${}^t\omega_{t-1}$ and ${}^tT_{t-1}$, for each "rigid" body ("rigid" from the viewpoint of the observer based on recent observation). The KF can be divided into two parts: updating and prediction. Before updating, the Kalman gain matrix should be estimated using the following equation:

$$K_t=P_{t|t-1}F_t^T(F_tP_{t|t-1}F_t^T+\Lambda_t)^{-1}, \quad \textbf{(20)}$$

where $P_{t|t-1}$ is the predicted state covariance matrix, which is extrapolated by the previous state covariance P_{t-1},

$$P_{t|t-1}=H_{t-1}P_{t-1}H_{t-1}^T+Q_{t-1}. \quad \textbf{(21)}$$

Then, the state vector s_t is updated using the measurement $x(t)$,

$$\hat{s}_t=\hat{s}_{t|t-1}+K_t(x_t-F_t\hat{s}_{t|t-1}), \quad \textbf{(22)}$$

and the state covariance matrix P_t is updated using

$$P_t=(I-K_tF_t)P_{t|t-1}. \quad \textbf{(23)}$$

The following equation can then be used to predict the state vector at time t+1 based on the measurement at time t:

$$\hat{s}_{t+1|t}=H\hat{s}_t. \quad \textbf{(24)}$$

Once we have the prediction of the state at the next time instant, $\hat{s}_{t+1|t}$, we can use Eq. (8) to compute the prediction of 3D feature location at time t+1:

$$\tilde{p}(t+1) = {}^{t+1}R_t[\hat{p}(t)-b(t)]+b(t+1), \quad \textbf{(25)}$$

where $b(t)$ is obtained from $\hat{s}_t$, $b(t+1)$ is obtained from $\hat{s}_{t+1|t}$, and ${}^{t+1}R_t$ can be computed from ${}^{t+1}\omega_t$ in $\hat{s}_{t+1|t}$ (i.e., the ω-components of $\hat{s}_{t+1|t}$) using the Rodrigues formula (Kanatani, 1990).

VI. Experimental Results

Figure 7 shows the active binocular head used in the experiments. This binocular head has eight degrees of freedom, controlled by the following eight motors: the left and right focus motors, the left and right vergence motors, the tilt motor, the pan motor and the X- and Y- motors. We have calibrated the binocular head using the method described proposed by Shih *et al*. (1998a, 1998b) and have achieved an accuracy of one pixel prediction error and 0.2 pixel epipolar error on average, even when all the joints are moved simultaneously. Since our binocular head is well calibrated, the inverse kinematics of the binocular head can be used to control the stereo cameras. The goal of inverse kinematics is to compute the joint values of the robot head such that the optical axes of both stereo cameras will pass through a given 3D target point. Eight constraints were provided to solve a unique set of joint values for controlling our binocular head. With the above capability, we could then easily control the binocular head so that it would fixate on the 3D point tracked by our 3D visual

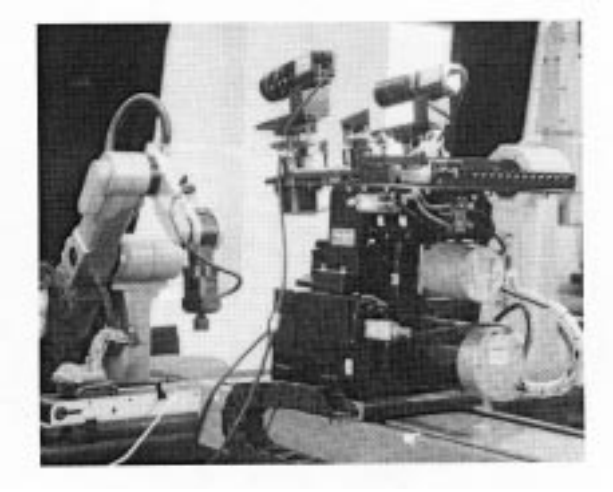


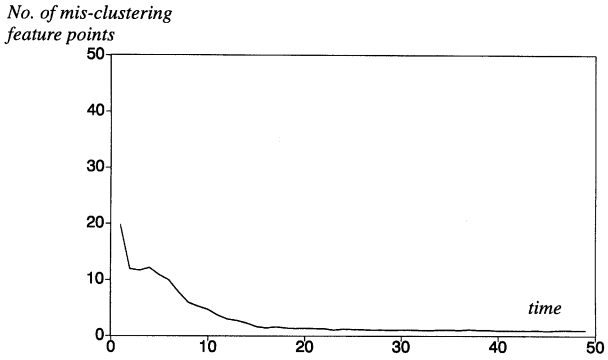
Fig. 7. The active binocular head used in our experiment.

tracker. For more details, please refer to Shih *et al.* (1998b).

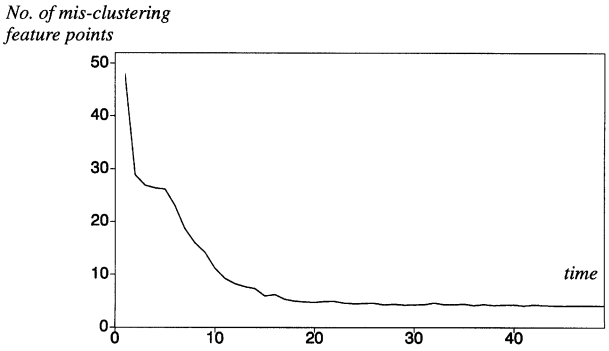
For our stereo vision system, the error in 3D point measurement, due to calibration inaccuracy and 2D feature detection error, was generally less than 2 millimeters. Therefore, *Tolerance_Tight* was set to 2 millimeters, which was the uncertainty of 3D point measurement. On the other hand, *Tolerance_Loose* was arbitrarily set to 5 millimeters in order to allow some non-rigidity or slight mis-correspondence. $N_{model\_required}$ was three because the Arun method requires at least three point correspondences to compute the motion parameters, and $N_{\min}$ was arbitrarily set to ten because a rigid body needed to have at least ten features in our experiments. In the following experiments, the window size for the similarity test used in the template matching module was 7×7 pixels, and the *consistency_tolerance* used to check the mutually-supported consistency was set to 2 pixels. Because of the sensitivity problem in estimating angular acceleration, we assumed constant angular velocity in our system, i.e., that the 3D object underwent a *constant* 3D rotation *during a time interval and* $\omega(t) = \omega(t-1)$.

We will present three sets of experimental results in this section. The first two are computer simulations, and the third one contains some experimental results on real image sequences. The first simulation shows the results of applying the RANSAC-based motion clustering, and the second one shows the results of applying KF. Each moving object, with both translational motion and rotational motion, contains 26 feature points on the surface of one rigid object (which was a cubic of 20 mm in our simulation). At each time instant, we computed the misclassification rate by counting the number of the feature points which were not assigned to the correct cluster (the correct members of each cluster were known in the simulation). The results of the simulation using the RANSAC-based motion clustering are shown in Fig. 8. As can be seen from Fig. 8, when the number of moving objects increased from three to five, the misclassification rate increased from 1/78 to 4/130, which was acceptable since all the five moving objects were still clustered correctly.

Once the feature points were clustered into a few clusters, each cluster was then assigned an individual KF. The synthetic data contained 26 feature points on the surface of one rigid object, just as in the previous simulation. Let $b(0)=(b_x,b_y,b_z)^T=(-100,150,-100)^T$ mm, $v=(v_x,v_y,v_z)^T=(5.0,0,0)^T$ mm/frame, $\omega=(\omega_x,\omega_y,\omega_z)^T=(0.02,0,0)^T$rad/frame, $a=0$ and $\mu=0$. Gaussian noise with zero mean and standard deviations of σ_x, σ_y and σ_z was added to the 3D coordinates of each feature point in the x, y, and z directions. Let $\hat{p}$ be the true 3D coordinates (without adding noise), and let $\tilde{p}$ be the predicted 3D coordinates. The 3D prediction error at time t is defined as



(a) Three moving objects (78 features).



(b) Five moving objects (130 features).

Fig. 8. The results with respect to the number of the mis-clustering feature points and that of objects are shown. Each experiment (three objects and five objects) was repeated (the repeat number was one hundred), and then the results ((a) and (b)) were obtained by averaging.

$$\frac{1}{N}\sum_{i=1}^{N}\left\|\hat{p}_i(t)-\tilde{p}_i(t)\right\|, \qquad \textbf{(26)}$$

where N=26 in this experiment. The noise level of the measurements shown in Fig. 9 was: $\sigma_x=\sigma_y=0.1$, $\sigma_z=0.2$. Note that the values of σ_x, σ_y and σ_z chosen in this simulation reflect the real situation of 3D stereo measurement with our setup. Figure 9(a)-(c) show the results of the state vector of our KF, which includes three rotation parameters, three rotation center parameters, and three velocity parameters. The 3D prediction error is shown in Fig. 9(d). The simulation shown in Fig. 10 used $\sigma_x=\sigma_y=0.2$, $\sigma_z=0.4$. Here, the offset of the rotation center b along the rotation axis could be set arbitrarily without affecting the motion because the direction of the rotation axis was fixed due

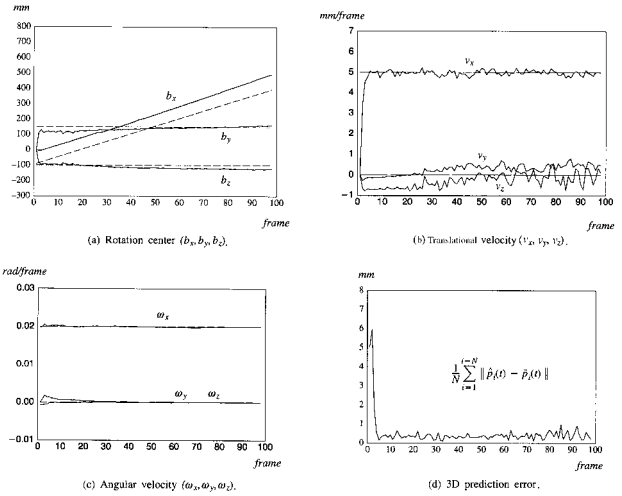


Fig. 9. The simulation results obtained using a Kalman filter (noise level for position was 0.1, 0.1, 0.2 mm).

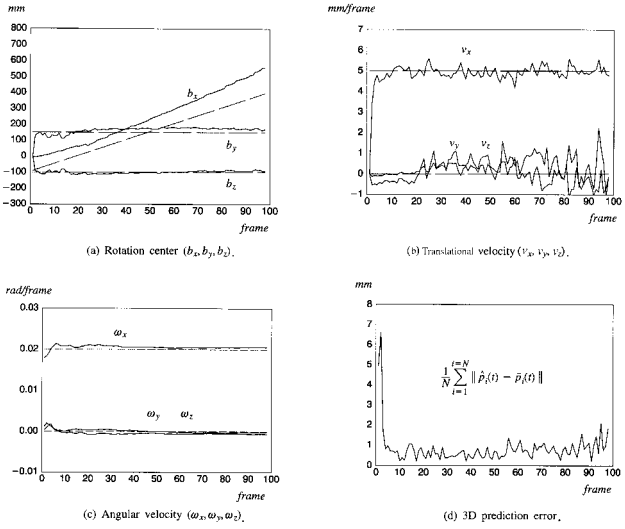


Fig. 10. The simulation results obtained using a Kalman filter (noise level for position was 0.2, 0.2, 0.4 mm).

to the use of zero angular acceleration in our simulation.

In the following, we will give the results of real experiments with four image sequences, each containing 30 stereo image pairs. In the first image sequence, a cola can was moving from right to left on a conveyor belt while the observer and the background objects were stationary. In the second image sequence, the cola can was moved from right to left as in the first image sequence, and the binocular head was panned (0.2° per frame) from right to left while the background was stationary. In the third image sequence, the background was still stationary, but the moving cola can was roughly fixated by the active binocular head such that the cola can was approximately held at the center of the image. In all the above three image sequences, the cola can move approximately 7 millimeters per frame. Figures 11 , 13 and 15 show the initial stereo correspondences of each image sequence, respectively. The image size is 512 by 512 pixels, and the corner features are marked by 5×5 squares. The initial stereo correspondences obtained with our automatic matching algorithm were quite reliable, which provided a good foundation for the subsequent tracking. There were 56~73 features on the cola can and 136~144 features on the background, depending on the image sequence, the time instant, and whether it was a left or right image. Figures

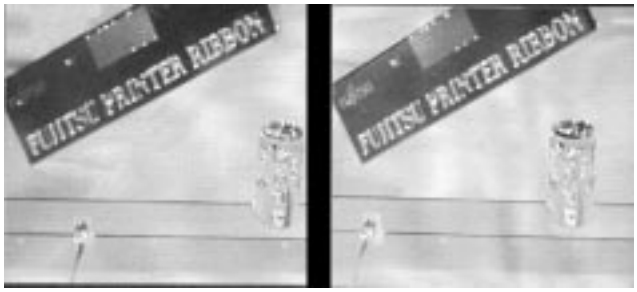


Fig. 11. (Image sequence 1) Initial stereo correspondence pairs superimposed on the first stereo image pair.

Fig. 12. (Image sequence 1) Trajectories of the tracked feature points superimposed on the last stereo image pair.

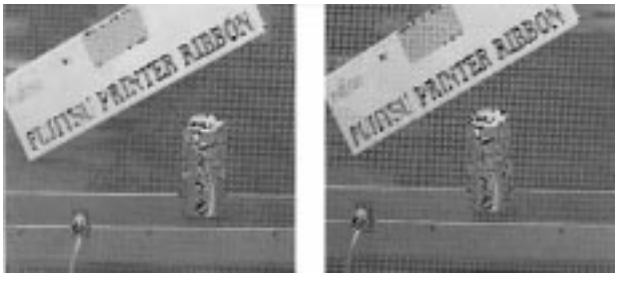


Fig. 13. (Image sequence 2) Initial stereo correspondence pairs superimposed on the first stereo image pair.

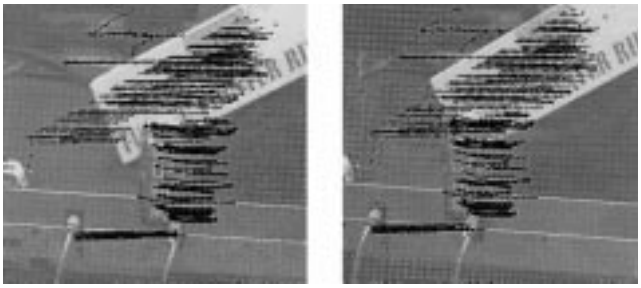


Fig. 14. (Image sequence 2) Trajectories of the tracked feature points superimposed on the last stereo image pair.

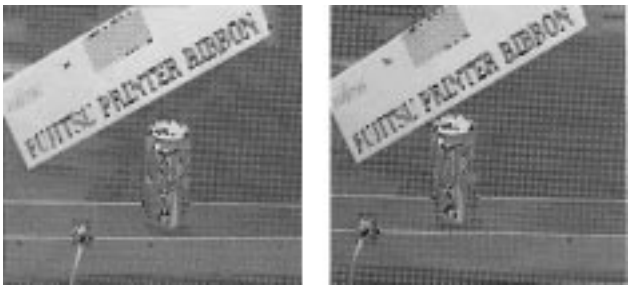


Fig. 15. (Image sequence 3) Initial stereo correspondence pairs superimposed on the first stereo image pair.

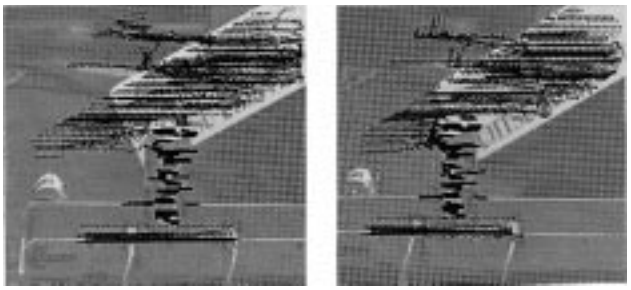


Fig. 16. (Image sequence 3) Trajectories of the tracked feature points superimposed on the last stereo image pair.

12, 14 and 16 show the trajectories of the tracked 3D features plotted on the last stereo image pair, where the 2D image locations of the 3D feature points at each time instant are marked with small crosses 5 pixels high. Square marks indicate the 2D locations, in the last image, of those feature points which were still in track after 30 image frames. In the last image sequence, two objects were moving with different motions while the background was stationary. Figure 17 shows the initial stereo correspondences of image sequence 4. The corresponding cluster and the trajectories for each tracked 3D feature plotted on the last stereo image pair are shown in Fig. 18. As can be seen in Fig. 18, the two moving objects with different motions and the still object (i.e., stationary background) are correctly clustered into three clusters. The above experimental results demonstrate that objects with different 3D motions can be successfully clustered and estimated using our algorithm.

VII. Conclusion

In order to record and analyze the dynamic scene automatically, an intelligent system should be able to fixate its cameras on a moving object so that the object can remain in the field of view. Therefore, we use a controlled binocular head, which allows us to track 3D moving objects using 3D approaches. That is, all the object features can be first reconstructed in the 3D space using stereo vision techniques and then clustered and tracked in the 3D space directly. The major advantages of using a binocular head are as follows: (1) The simple and fast linear KF instead of the more complicated EKF can be used. To observe in 2D images and track in 3D space is the goal of visual tracking. The relationship between 2D images and 3D features is a perspective transformation which is inherently nonlinear. Therefore, the formulation of a motion model involving perspective transformation becomes

Fig. 17. (Image sequence 4) Initial stereo correspondence pairs superimposed on the first stereo image pair.

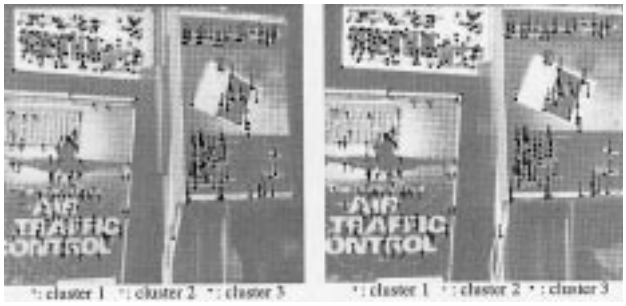


Fig. 18. (Image sequence 4) Clusters and trajectories of the tracked feature points superimposed on the last stereo image pair.

nonlinear, and EKF is usually adopted in the visual tracking task which uses KF techniques. On the other hand, our 3D approach can lead to a linear formulation of motion model, thus allowing use of a faster and simpler linear KF. (2) Occlusion can be easily detected or even predicted. If the 3D structure and motion of each object can be made available, occlusion may be easily detected or predicted in advance. With our 3D approach, the 3D structure and motion of each moving object can be estimated. Therefore, the occlusion problem, which is one of the most difficult problems in computer vision, can be dealt with more easily. (3) A 3D model can be built on line and then utilized in model-based visual tracking. With our 3D approach which utilizes a binocular head, it is not necessary to build the 3D model in advance. We are currently working on model-based face tracking using this approach. Using the binocular head, we are able to simultaneously reconstruct and track a moving object (or more specifically, a moving face).

This paper has presented a 3D feature-based tracker capable of tracking multiple moving objects even when the stereo camera set is moving. Therefore, it can be used to control the binocular head (a robot manipulator for moving cameras around) so that it will fixate its cameras on the object in which it is interested. This tracker is completely autonomous in the sense that it requires no initial correspondence of any kind, either temporal or stereo correspondence. Any 3D rigid objects, or any articulated object such as a robot manipulator, can be tracked using our tracker as long as there are some corner features on each moving component.

The performance of our 3D tracker is mainly based on the following features: (1) A RANSAC-based motion clustering and estimation algorithm using the rigid body consensus has been proposed for grouping features into common-motion clusters and estimating their 3D motion simultaneously. This clustering method provides a systematic way of managing splitting, merging, new appearance and disappearance of multiple moving rigid objects. (2) Linear KFs are used to predict the next movements of common-motion clusters, which can then be used to provide better prediction for 3D feature tracking (i.e., finding temporal correspondence). (3) Two parallel 2D temporal modules are utilized to make full use of the temporal information contained in the image sequence. (4) The constraint of mutually-supported consistency has been introduced to eliminate incorrect stereo correspondences. (5) Calibration is done automatically and accurately to assure the accuracy of 3D inference. Preliminary experiments have shown that our tracking system does give good results and can serve as a robust 3D feature tracker for our active binocular vision system.

Acknowledgment

We would like to thank Dr. Leslie Kitchen for his help with implementation of the feature extraction module and for some discussions on system design. We want to thank Dr. Lin-Guo Liou for useful suggestions on motion modeling and tracking. This work was supported in part by the National Science Council, R.O.C., under grants NSC 85-2213-E-001-006 and NSC 86-2745-E-001-007.

Appendix 1

This appendix contains Algorithm 1, which is used to extract corner features and was originally developed by Cooper *et al.* (1993). This algorithm extracts a set of corner features from an intensity image.

Algorithm 1. Feature Extraction:
Given an intensity image, extract a set of corner features.
For each pixel (x,y) in an image:
Step 1: Compute $\nabla_x \mathbf{I}$ and $\nabla_y \mathbf{I}$ (e.g., using the Sobel operator).
Step 2: **IF** $|\nabla_x \mathbf{I}|+|\nabla_y \mathbf{I}| \le$ Threshold_Gradient, **THEN**
This point (x, y) is not a corner.
ELSE
Calculate positions (x_l,y_l) and (x_r,y_r) to the left and right along the local contour direction. Use a similarity test to decide whether the image patch centered at (x_l,y_l) or at (x_r,y_r) differs from that at (x,y).
Step 2.1: **IF** they differ, **THEN**
(x,y) is a corner.
ELSE
(x,y) is not a corner.
END of Feature Extraction

Appendix 2

This appendix contains two algorithms, Algorithm 2 and Algorithm 3, which are used to find the stereo correspondence. Two constraints are used in finding the stereo correspondence: the epipolar-line constraint is used to restrict the search region when finding stereo correspondence using template matching, and the mutually-supported consistency constraint is used to eliminate unreliable stereo correspondence. In our tracking system, only mutually-supported stereo correspondence pairs are used to compute 3D features using triangulation.

Algorithm 2. Initial Stereo Correspondence:
Given two sets of salient features, one in the left image and one in right image, at time t =0, this algorithm determines the initial stereo correspondence pairs.
Step 1: For each suppressor feature in the left image:
Step 1.1: Determine the corresponding search area in the right image using the epipolar line constraint.
Step 1.2: For each salient feature (suppressor or suppressed) falling in the search area, compute the similarity measure $\boldsymbol{J}$.
Step 1.3: The feature having the largest $\boldsymbol{J}$ is regarded as a *possible* corresponding feature in the right image.
Step 2: For each suppressor feature in the right image: repeat the procedure similar to the one in **Step 1**.

Step 3: Use the mutually-supported consistency constraint to eliminate the unreliable stereo correspondence found in **Step 1**.

END of Initial Stereo Correspondence

Algorithm 3. Stereo Correspondence and Triangulation:

For each 3D feature in track, this algorithm attempts to find a unique stereo pair from among the multiple candidates of temporal correspondence in both images, which are obtained from the left and right 2D temporal matching modules, and it then computes an estimate of an 3D feature location using stereo triangulation.

For each 3D feature in track:

Step 1: For each temporal matching candidate $q^L(t,k)$, k=1, 2, ..., N_c, in the left image:

Step 1.1: Determine the corresponding search area in the right image using the epipolar line constraint.

Step 1.2: For each temporal matching candidate $q^R(t,k)$ falling in the search region, within the 3×3 search window centered at $q^R(t,k)$ (that is, we check not only the feature $q^R(t,k)$, but also its eight neighbors in order to correct the small localization error caused by the previous temporal matching module), find the best stereo match by computing the similarity measure **J** with the intensity pattern centered at $q^L(t,k)$.

Step 2: For each temporal matching candidate $q^R(t,k)$, k=1, 2, ..., N_c, in the right image, repeat the procedure used in **Step 1**.

Step 3: Among the stereo matches obtained in **Steps 1** and **2**, use the mutually-supported consistency constraint to choose the best stereo match in the following way:

Step 3.1: **IF** there is no mutually-supported consistency pair, **THEN**
do not track this 3D feature for this time instant.

Step 3.2: **IF** there is only one mutually-supported consistency pair, **THEN**
choose this pair as the best stereo pair for the 3D feature in track.
ELSE {more than one mutually-supported consistent pair}
choose the stereo pair having the minimum error in 2D temporal matches and the largest corner strength in feature detection.

Step 4: With the stereo correspondence found in **Step 3**, compute an estimate of the 3D feature location $\hat{p}$ (t) using stereo triangulation.

END of Stereo Correspondence and Triangulation

Appendix 3

This appendix contains Algorithm 4, which is used to find the temporal correspondence. This algorithm generates a set of candidates of temporal correspondence, which are the input for stereo correspondence.

Algorithm 4. 2D Temporal Matching:

Given a set of 2D features $\{q'^L(t)\}$ (or $\{q'^R(t)\}$) obtained in **Step 2.1** in the **Main Algorithm**, this algorithm generates a set of candidates of temporal correspondence $\{q^L(t,k), k=1, 2, ..., N_c\}$ (or $\{q^R(t,k), k=1, 2, ..., N_c\}$) for each 2D feature prediction $\tilde{q}^L(t)$ (or $\tilde{q}^R(t)$) associated with a 3D feature. (In the following description, we use the left image as an example.)

For each of the 3D features in track:

Step 1: For each $q'^L(t)$ falling in the search region centered at $\tilde{q}^L(t)$, within the 3×3 window centered at $q'^L(t)$ (that is, we check not only the feature $q'^L(t)$, but also its eight neighbors in order to correct the small localization error of $q'^L(t)$ caused by the feature extraction module), find the best temporal match using a similarity test with the intensity pattern, associated with the 3D feature, in the previous image $I^L(t-1)$.

Step 2: Among the matches found in **Step 1**, choose the N_c best temporal matches as candidates of the image location corresponding to the 3D feature in track.

END of 2D Temporal Matching

Appendix 4

This appendix contains four algorithms, Algorithm 5 – Algorithm 8, which are used to perform 3D motion clustering. In the initial clustering stage, Algorithm 5 partitions 3D feature points into com- mon-motion clusters. After that, Algorithm 8 is used to maintain the appearance, disappearance, splitting, and merging of the active common-motion clusters of 3D features. Algorithm 6 and Algorithm 7 describe two functions which will be called by Algorithm 5 and Algorithm 8.

Algorithm 5. Initial Clustering:

Given a set of 3D correspondences between t=0 and t=1, $S(0,1)$, this algorithm partitions the 3D feature points into several clusters, each having a common 3D motion, and possibly, one un-clustered set S^0.

Step 1: Given $S(0,1)$, partition 3D feature points into several clusters using **Algorithm 6**. Those 3D features classified as belonging to a common-motion cluster in this step are *members* (in contrast to *candidates*) of that cluster.

Step 2: For each 3D feature point that was not classified into any common-motion cluster in **Step 1**, check whether it can be assigned as a *candidate* of a common-motion cluster in the following way:
The ith feature point, with 3D correspondence $(\hat{p}_i(0), \hat{p}_i(1))$, is classified into cluster j if it gives the smallest $e_i^j = \|\hat{p}_i(1) - {}^1R_0^j \hat{p}_i(0) - {}^1T_0^j\|$ for all j and $e_i^j \le Tolerance\_Loose$, where ${}^1R_0^j$ and ${}^1T_0^j$ are the rotation matrix and the translation vector, respectively; otherwise, assign this feature to be an un-clustered feature.

END of Initial Clustering

Algorithm 6. Motion Clustering of an Un-clustered Set:

Given an un-clustered set of 3D correspondences over time, S^0, this algorithm partitions the 3D feature points into several common-motion clusters and, possibly, one remaining un-clustered set.

Repeat the following procedure until $\#(S^0) < N_{\min}$:

Step 1: Find the largest rigid body consensus set $S^* \subseteq S^0$ and its common 3D motion ${}^tR_{t-1}$ and ${}^tT_{t-1}$ using **Algorithm 7**.

Step 2: **IF** $\#(S^*) \ge N_{\min}$, **THEN**

Step 2.1: Construct a new common-motion cluster with a unique cluster identification having the motion ${}^tR_{t-1}$ and ${}^tT_{t-1}$. The 3D feature point whose 3D correspondence pair is in S^* will be assigned as a *member* of this new common-motion cluster.

Step 2.2: Remove all the elements of S^* from S^0.

ELSE
Break from the Repeat procedure.

END of Motion Clustering of an Un-clustered Set

Algorithm 7. Finding the Largest Rigid Body Consensus Set:

Given a set of 3D correspondences over time, S, this algorithm finds the largest rigid body consensus set and its common 3D motion.

Step 1: Determine the number of random trials, $N_{trial}=\frac{k}{w^3}$ (refer to Fischler and Bolles (1981)).

Step 2: Find a motion consensus set instantiated by a random sample of size three from S:

Step 2.1: Randomly select a subset $S1$ of three feature pairs from S and compute the motion ${}^{t}R_{t-1}$ and ${}^{t}T_{t-1}$ using the Arun method (Arun *et al*., 1987).

Step 2.2: Form the consensus set $S1^*$ from S, where $S1^*$ is the set of all $(\hat{p}_i(t-1), \hat{p}_i(t)) \in S$ such that $\|\hat{p}_i(t) - {}^{t}R_{t-1}\,\hat{p}_i(t-1) - {}^{t}T_{t-1}\| < Tolerance\_Tight$.

Step 3: Repeat **Step 2** for N_{trial} times, and keep the largest consensus set.

Step 4: Recompute the motion parameters ${}^{t}R_{t-1}$ and ${}^{t}T_{t-1}$ from the largest consensus set using a weighted version of the Arun method.

END of Finding the Largest Rigid Body Consensus Set

Algorithm 8. Cluster Maintenance:

This algorithm maintains the appearance, disappearance, splitting, and merging of the active common-motion clusters of 3D features.

Step 1: For each common-motion cluster j, j=1, 2, ..., $L(t-1)$:

Step 1.1: Find the largest rigid body consensus set $A \subseteq S_M^j(t-1,t)$, and update the motion parameters ${}^{t}R_{t-1}^{j}$ and ${}^{t}T_{t-1}^{j}$ using **Algorithm 7**.

Step 1.2: **IF** #(A) is smaller than $N_{model\_required}$(=3), *the minimum number of 3D feature pairs required to instantiate a model,*

THEN

Designate all its members and candidates as un-clustered, and delete this cluster (i.e., this common-motion cluster disappears).

ELSE

Step 1.2.1: Move the 3D correspondences which are in $S_M^j(t-1,t) \backslash A$ from $S_M^j(t-1,t)$ to the un-clustered set S^0. That is, the inconsistent *members* are removed from the jth cluster and designated as un-clustered.

Step 1.2.2: For each candidate in the jth common-motion cluster, check whether it can be promoted to be a *member* in the following way:

If its motion similarity error with respect to the jth cluster is smaller than *Tolerance_Tight*, then promote it to be a member; otherwise, remove it from the candidate set of the jth cluster, and designate it as an un-clustered feature, for the moment; it may then become a candidate of an active common-motion cluster in **Step 2** or a member of a new cluster in **Step 3**.

Step 2: For each un-clustered 3D feature, check whether it can be designated as a candidate of an active common-motion cluster in the following way:

If its motion similarity error with respect to the most similar-in-motion cluster is less than *Tolerance_Loose*, then designate it as a *candidate* of that most similar-in-motion cluster; otherwise, keep it un-clustered.

Step 3: If #(S^0) is greater than $N_{\min}$, then apply **Algorithm 6** to S^0 to find new common-motion clusters (at this moment, with members only).

Step 4: For each active common-motion cluster j:

Step 4.1: If there is a cluster k having a 3D motion in common with cluster j for three consecutive cycles, then merge these two clusters.

Step 4.2: If the number of its members is less than $N_{\min}$ for three consecutive cycles, then designate all its members and candidates as un-clustered, and delete this cluster (i.e., this common-motion cluster disappears).

End of Cluster Maintenance

References

Ahuja, N. and A. L. Abbott (1993) Active stereo: integrating disparity, vergence, focus, aperture, and calibration for surface estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **15**(10), 1007-1029.

Allen, P. K., A. Timcenko, B. Yoshimi, and P. Michelman (1993) Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Trans. Robotics and Automation*, **9**(2), 152-165.

Aloimonos, J. Y., I. Weiss, and A. Bandopadhay (1987) Active vision. *Proceedings of* 1*st Int. Conf. on Computer Vision*, pp. 35-54. London, U.K.

Arun, K. S., T. S. Huang, and S.D. Blostein (1987) Least-square fitting of two 3-D point sets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **9**(5), 698-700.

Azarbayejani, A. and A. Pentland (1995) Recursive estimation of motion, structure, and focal length. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **17**(6), 562-575.

Bajcsy, R. (1988) Active perception. *Proc. IEEE*, **76**(8), 996-1005.

Bar-Shalom, Y. and T. E. Fortmann (1988) *Tracking and Data Association*. Academic, San Diego, CA, U.S.A.

Basri, R. and S. Ullman (1988) The alignment of objects with smooth surfaces. *Proceedings of* 2*nd Int. Conf. on Computer Vision*, pp. 482-488. Tampa, FL, U.S.A.

Bradshaw, K. J., P. F. McLauchlan, I. D. Reid, and D. W. Murray (1994) Saccade and pursuit on an active head/eye platform. *Image and Vision Computing*, **12**(3), 155-163.

Broida, T. J., S. Chandrashekhar, and R. Chellappa (1990) Recursive 3-D motion estimation from a monocular image sequence. *IEEE Trans. Aerospace and Electronic Systems*, **26**(4), 639-656.

Cheng, T. K. and L. Kitchen (1993) *Multi-Agent 3D Tracking and Segmentation Using Stereo.* Technical Report 1993/30, Department of Computer Science, University of Melbourne, Parkville, Victoria, Australia.

Christensen, H. (1993) A low-cost robot camera head. *Int. J. Pattern Recogn. Artif. Intell.*, **7**(1), 69-87.

Christensen, H., J. Horstmann, and T. Rasmussen (1995) A control theoretical approach to active vision. *2nd Asian Conf. on Computer Vision (Recent Developments in Computer Vision: invited session papers)*, pp. 201-210. Singapore.

Clarke, J. C. and A. Zisserman (1996) Detection and tracking of independent motion. *Image Vision Computing*, **14**, 565-572.

Cooper, J. (1994) *Real-Time Task-Directed Robot Vision*. Ph.D. Dissertation. Department of Computer Science, University of Western Australia, Crawley WA, Australia.

Cooper, J., S. Venkatesh, and L. Kitchen (1993) Early jump-out corner detectors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **15**(8), 823-828.

Crowley, J., P. Bobet, and M. Mesrabi (1992) Gaze control for a binocular camera head., *Proceedings SPIE: Applications of Artificial Intelligence X: Machine Vision and Robotics*, Vol.

1708, pp. 47-61. Orlando, FL, U.S.A.

Faugeras, O. (1993) *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, Cambridge, MA, U.S.A. and London, U.K.

Ferrier, N. J. and J. J. Clark (1993) The Harvard binocular head. *Int. J. Pattern Recogn. Artif. Intell.*, **7**(1), 9-31.

Fiala, J. C., R. Lumia, K. J. Roberts, and A. J. Wavering (1994) Triclops: a tool for studying active vision. *Int. J. Computer Vision*, **12**(2/3), 231-250.

Fischler, M. A. and R. C. Bolles (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24**(6), 381-395.

Gee, A. and R. Cipolla (1996) Fast visual tracking by temporal consensus. *Image Vision Computing*, **14**, 105-114.

Huttenlocher, D. P. and S. Ullman (1990) Recognizing solid objects by alignment with an image. *Int. J. Comput. Vision*, **5**(2),195-212.

Kanatani, K. (1990) *Group-Theoretical Methods in Image Understanding*. Springer-Verlag, Berlin and Heidelberg, Germany.

Koller, D., K. Daniilidis, and H. H. Nagel (1993) Model-based object tracking in monocular image sequences of road traffic scenes. *Int. J. Computer Vision*, **10**(3), 257-281.

Krotkov, E. P. and R. Bajcsy (1993) Active vision for reliable ranging: cooperative focus, stereo and vergence. *Int. J. Computer Vision*, **11**(2), 187-203.

Lew, M. S., T. S. Huang, and K. Wong (1994) Learning and feature selection in stereo matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **16**(9), 869-881.

Meer, P., D. Mintz, and A. Rosenfeld (1991) Robust regression methods for computer vision: a review. *Int. J. Comput. Vision*, **6**(1), 59-70.

Milios, E., M. Jenkin, and J. Tsotsos (1992) TRISH: a binocular robot head with torsional eye movements. *Proceedings SPIE: Applications of Artificial Intelligence X: Machine Vision and Robotics*, Vol. 1708, pp. 36-46. Orlando, FL, U.S.A.

Murray, D. M., K. J. Bradshaw, P. F. McLaulan, I. D. Reid, and P. M. Sharkey (1995) Driving Saccade to pursuit using image motion. *Int. J. Computer Vision*, **16**(3), 205-228.

Pahlavan, K. and J. O. Eklundh (1992) A hand-eye system-analysis and design. *Comput. Vision Graph. Image Process.: Image Understanding*, **56**(1), 41-56.

Papanikolopoulos, N. P. (1992) *Controlled Active Vision*. Ph.D. Dissertation. Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, U.S.A.

Papanikolopoulos, N. P., B. Nelson, and P. K. Khosla (1992) Monocular 3-D tracking of a moving target by eye-in-hand robotic system. *Proceedings of* 31*th Conf. on Decision and Control*, pp. 3805-3810. Tucson, AR, U.S.A.

Reid, I. D. and D. W. Murray (1996) Active tracking of foveated feature clusters using affine structure. *Int. J. Computer Vision*, **18**(1), 41-60.

Roth, G. and M. Levine (1993) Extracting geometric primitives. *CVGIP: Image Understanding*, **58**(1), 1-22.

Rousseeuw, P. J. and A. M. Leroy (1987) *Robust Regression & Outlier Detection*. Wiley, New York, NY, U.S.A.

Sharkey, P. M., D. W. Murray, S. Vandevelde, I. D. Reid, and P. F. McLauchlan (1993) A module head/eye platform for real-time reactive vision. *Mechatronics*, **3**(4), 517-535.

Shih, S. W., Y. P. Hung, and W. S. Lin (1998a) New closed-form solution for Kinematic parameter identification of a binocular head using stereo point measurements. *IEEE Trans. Systems, Man, and Cybernetics*, **28B**(2), (to appear).

Shih, S. W., Y. P. Hung, and W. S. Lin (1998b) Calibration of an active binocular head. *IEEE Trans. Systems, Man, and Cybernetics*, **28A**(3), (to appear).

Shoham, D. and S. Ullman (1988) Aligning a model to an image using minimal information. *Proceedings of 2nd Int. Conf. on Computer Vision*, pp. 259-263. Tampa, FL, U.S.A.

Ullman, S. and R. Basri (1991) Recognition by linear combinations of models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **13**(10), 992-1006.

Weng, J., T. Huang, and N. Ahuja (1987) 3-D motion estimation, understanding, and prediction from noisy image sequences. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **9**(3), 370-389.

Yang, C. H. and J. J. Leou (1993) Robot tracking by stereo vision. *Proceedings of Conf. on Computer Vision, Graph. and Image Processing*, pp. 255-262. Nantou, Taiwan, R.O.C.

Yao, Y. S. and R. Chellappa (1995) Tracking a dynamic set of feature points. *IEEE Trans. Image Processing*, **4**(10), 1382-1395.

Young, G. S. and R. Chellappa (1990) 3-D motion estimation using a sequence of noisy stereo images: models, estimation, and uniqueness results. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**(8), 735-759.

Zhang, Z. and O. Faugeras (1992a) *3D Dynamic Scene Analysis: a Stereo Based Approach*. Springer-Verlag, Berlin and Heidelberg, Germany.

Zhang, Z. and O. Faugeras (1992b) Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *Int. J. Comput. Vision*, **7**(3), 211-241.

Zhang, Z., R. Deriche, O. Faugeras, and Q. T. Luong (1995) A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, **78**, 87-119.

Zheng, Q. and R. Chellappa (1995) Automatic feature point extraction and tracking in image sequence for arbitrary camera motion. *Int. J. Comput. Vision*, **15**, 31-76.

可追蹤多個物體之基於特徵的三維追蹤器

唐政元[*,**]　洪一平[**]　石勝文[***]　陳　稔[*]

[*]國立交通大學資訊工程研究所
[**]中央研究院資訊科學研究所
[***]國立暨南大學資訊工程系

摘　要

在此論文中，我們使用一個可控制的雙眼機械頭追蹤多個運動物體之基於特徵的三維追蹤器。我們的追蹤器之操作分成兩個階段：初始階段與追蹤階段。在初始階段，我們使用同軸線限制與互相支援的一致性來可靠地決定在第一個立體影像配對中的立體對應。在追蹤階段，我們建立一個回饋迴路：首先使用卡氏濾波器預測新的三維位置，然後投影到新的影像來引導在新的影像之特徵抽取。在本論文中，我們使用剛體一致性的原理（所有在同一個剛體上所抽取出的特徵擁有相同的運動），提出一個基於RANSAC的分群法來作運動分割與估測。這個新的特徵分群演算法提供了一個系統化的方法來管理多個運動物體（包含連接的物體，如機械人的操作手臂）的分離、結合、出現與消失。因為我們使用基於RANSAC的方法所找到的運動估測當作是卡氏濾波器的量測值，所以可以使用線性的卡氏濾波器取代常被使用的非線性的卡氏濾波器來作預測視覺追蹤。實驗顯示出我們的追蹤系統所獲得的追蹤結果相當好，將可提供我們的主動雙眼機械頭一個準確度很高的三維特徵追蹤器。