A 3D Feature-Based Tracker for Tracking Multiple Moving Objects with a Controlled Binocular Head

Yi-Ping Hung[1], Cheng-Yuan Tang[2], Sheng-Wen Shih[3], Zen Chen[2], Wei-Song Lin[3]

[1]Institute of Information Science, Academia Sinica, Taipei, Taiwan

[2]Institute of Computer Science & Information Engineering,
National Chiao Tung University, Hsinchu, Taiwan

[3]Institute of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Email: hung@iis.sinica.edu.tw

ABSTRACT

Object tracking is an important task for active vision and robotics. This paper presents a 3D feature-based tracker for tracking multiple moving objects with 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 by using the epipolar line constraint and the *mutually-supported consistency.* In the tracking phase, the feedback loop is established by first predicting new 3D feature locations with the Kalman filters 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-based clustering method for motion segmentation and estimation by using the principle of *rigid body consensus* which states that all the extracted 3D features on a rigid body have the same 3D motion. This new method leads to a feature-clustering algorithm which provides a systematic way for managing splitting, merging, appearance and disappearance of multiple moving rigid objects — including articulated objects, such as robot manipulators. By using the motion estimates obtained with the RANSAC-based method as the measurements for the Kalman filters, we are able to use *linear* Kalman filters for predictive visual tracking, instead of the extended Kalman filters which most people used for tracking. Experiments have shown that our tracking system does give good results and can serve as a robust 3D feature tracker for an active binocular vision system.

I. INTRODUCTION

Advocated and pioneered by Aloimonos and Bajcsy in the eighties [3][5], 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 the cameras under computer control [1][7][10][11][12][17][18][19]. 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 the experimental equipment, active vision can be expected to receive even more attention from the researchers in the field 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 tracker for tracking multiple moving objects in a cluttered environment with a computer-controlled binocular head.

Our work is closely related to the work by Kitchen and Cooper. Cooper built a 2D feature-based tracker that can track about 35 feature points in the image at a rate of 5 frames per second and organized them into clusters on the basis of their motion [9]. 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 extended this 2D tracker to a 3D feature-based tracking system by utilizing information in stereo correspondence [6]. 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 is the tracking system built by Zhang and Faugeras [27][28], 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 its applications to classes of images where line features are prevalent [14]. In this paper, we deal only with point features. However, our approach can be easily extended to using both point features and line features.

Among a few other interesting works on visual tracking are the following. Wavering and Lumia used TRICLOPS, the binocular head they built at the NIST, to track an object undergoing random or periodic motion [12]. But only experiments on one moving object with simple background were shown. Allen *et al.* [2] used a stationary binocular system to track a single object in real time. Once tracking was stable, the system can 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.* [20][21] 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. Their vision system detected motion by computing optical flow based on the Sum of Squared Differences (SSD) method. Another interesting work on monocular visual tracking is the one by Koller, Daniilidis and Nagel [16], where 3D parameterized models were used for 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 with a monocular image sequence [16][21], many researchers who worked on monocular visual tracking before have now shifted to work on binocular visual tracking, e.g., Kitchen [6] and Leou [25]. The reason is simple. It is easier to "track" (and predict) 3D objects in 3D space rather than in 2D image domain. The motion segmentation problem becomes easier and occlusion can be easily detected or predicted by using 3D object and motion models. 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 to be a difficult problem. Therefore, many papers on binocular visual tracking and motion estimation assumed that stereo correspondence and/or 3D fea-

ture correspondence over the frames were given [24], either for every stereo pair [26] or for at least the first stereo pair [6]. 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 constraint of *mutually-supported consistency,* which can remove most of the false matches that happen to satisfy the epipolar line constraint.

The use of the Kalman filters (KF) or the extended Kalman filters (EKF) has now become popular for visual tracking. Young and Chellappa [26] described the computer simulation of a tracking system using the EKF that took a number of noisy 3D points assumed to belong to the same rigid object to estimate its motion. Zhang and Faugeras [27] derived some closed-form solutions for some 3D motion models, and used them to formulate an EKF to deal with nonlinear measurement equations. In this paper, we formulate a *linear* Kalman filter for motion tracking by using the motion estimates as the measurements for the Kalman filter.

In order to track multiple moving objects with Kalman filters, the system has to partition the 3D feature points into several common-motion clusters before applying Kalman filtering. Here, we encounter a famous dilemma. That is, if we do not know which feature belongs to which object beforehand, then we can not determine the motion for that object. However, if we do not know the motion for 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 by 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 [13]. 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 based on only short term 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 during a short time interval, J2 moved but J1 did not. Then, from the viewpoint of the observer, A & B form a rigid body together, while C is a different rigid body undergoing a different motion.

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 Kalman filtering. This paper is organized as follows. The system overview and the main algorithm are stated in section II. The temporal and stereo correspondence algorithms are described in section III. Motion clustering and estimation are described in section IV. Motion kinematics and the formulation for the Kalman filter are described in section V. Experimental results are shown in section VI. Conclusions are given in section VII.

II. SYSTEM OVERVIEW OF THE STEREO TRACKER

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 Figure 2 shows the system block diagram of the tracking phase. Initially, the stereo cameras are calibrated, in an automatic way, to have a 3D measurement accuracy of 1 millimeter (assuming good feature detection)

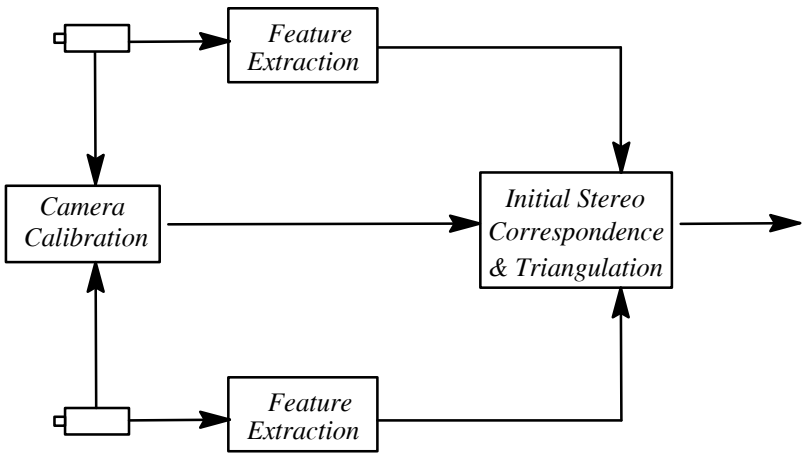

Figure 1. The system block diagram of the initialization phase.

[22]. 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 by 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 ambiguous and error–prone matches. Once the 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 problem simultaneously. By using the estimate of the successive motion obtained from the motion clustering and estimation module as the measurements of the Kalman filter, we can predict the 3D positions of the features at the next time instant, which are then projected onto the 2D image planes to provide the 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 searching time. 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 by using **Algorithm 1** with a larger threshold to allow only salient features.

Step 1.2: Solve the initial stereo correspondence problem by using **Algorithm 2**, and compute the initial estimates of the 3D feature locations by stereo triangulation.

Step 2: At time t = 1, 2, ... {the tracking phase}

For each new image pair

Step 2.1: Extract 2D features by 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 $\hat{p}(0)$ obtained 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 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 by using **Algorithm 4**.

Step 2.6: Use the RANSAC-based clustering method to segment multiple moving objects and estimate their 3D motion simultaneously:

IF t = 1, {The initial clustering stage}

THEN

Use **Algorithm 5** 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 by using **Algorithm 8**.

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

END of Main Algorithm

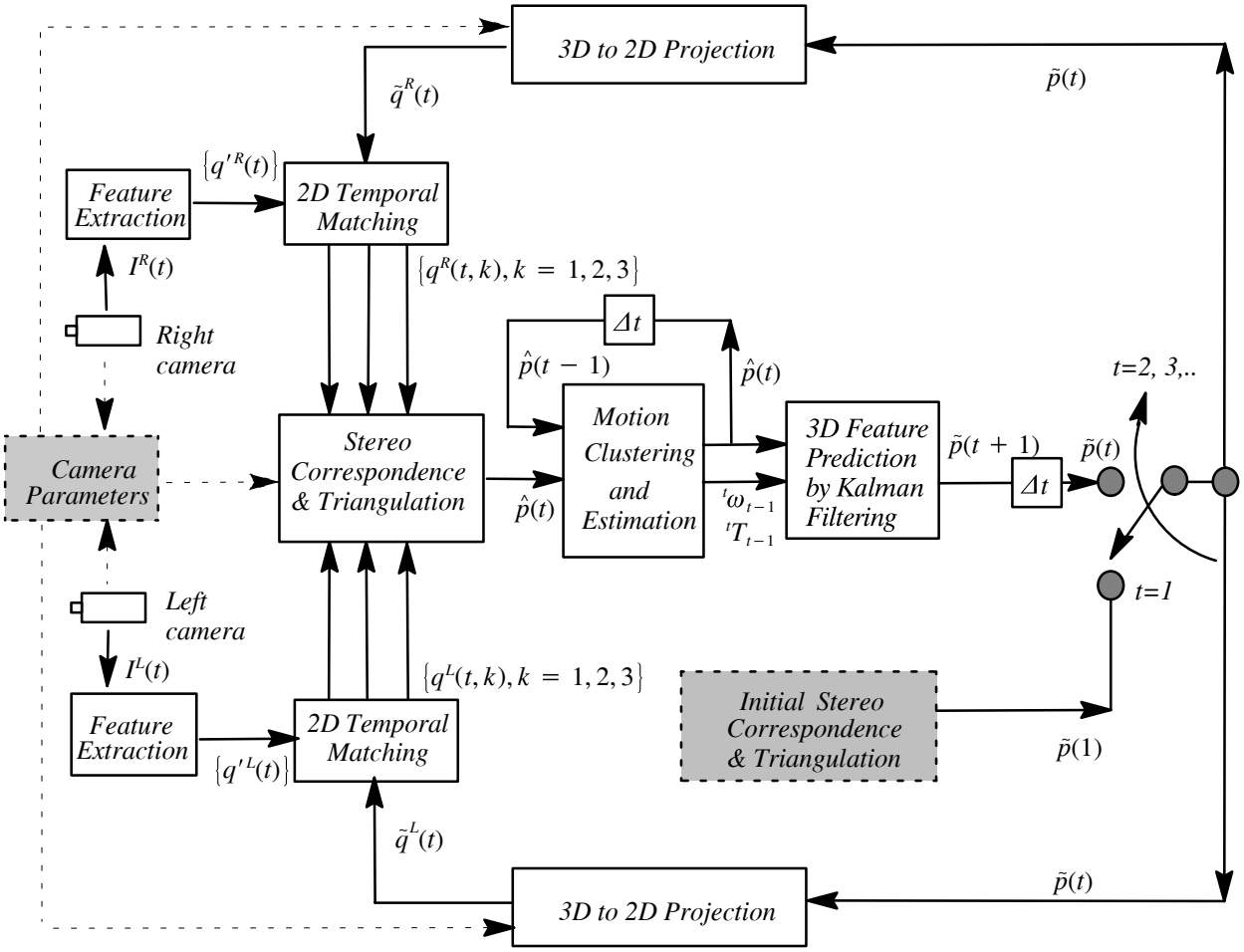

Figure 2. The system block diagram of the tracking phase ($t = 1, 2, ...$). The notation used is described below: $\hat{p}(t)$ is the estimate of the 3D coordinates of a 3D feature point P in track at time t. ${}^t\omega_{t-1}$ and ${}^tT_{t-1}$ are the estimates of the rotational and translational motion 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. $\{q^L(t,k), k = 1,2,3\}$ and $\{q^R(t,k), k = 1,2,3\}$ are the candidate sets of temporal correspondence in the left and right images, respectively. $\hat{p}(t)$ is the estimate of the coordinates of the 3D feature point P at time t.

III. TEMPORAL AND STEREO CORRESPONDENCE

A slightly modified version of the corner detector developed by Cooper, Venkatesh, and Kitchen [8] was used for extracting salient features. The basic algorithm for the feature extraction module is described below.

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** $|\nabla_x \mathbf{I}| + |\nabla_y \mathbf{I}| \leq$ Threshold_Gradient, **THEN**

This point (x, y) is not a corner.

ELSE

Calculate positions (x_l, x_l) and (x_r, x_r) to left and right along the local contour direction. Use a similarity test to decide whether the image patch centered at (x_l, x_l) or at (x_r, x_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

The above feature extraction method (as with most corner detectors) obtains multiple responses in the neighborhood of corner points; therefore a form of "non-maximum suppression" (on the basis of proximity) is used to eliminate the redundant feature points. With our algorithm, a corner feature is accepted as a *distinguished feature* if it is the strongest corner within its suppression reign (5×5 window in our experiments). Otherwise, it is suppressed by a distinguished feature and will be referred to as a *suppressed feature*. Notice that both the distinguished features and the suppressed features are *salient features* (i.e., obtained with a larger threshold in feature extraction) and will be used in the stereo correspondence module, but in a different way. For each feature point, the graylevel values in a small neighborhood are stored as a graylevel template for future matching.

III.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 there is no occlusion. Therefore, we can determine if the two feature points, q^L and q^R, are a stereo pair by checking if the image patches centered at q^L and q^R are similar. A popular similarity measure used for finding stereo correspondence is the sum of square difference:

$$\boldsymbol{J} = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} [\boldsymbol{I}^L(i,j) - \boldsymbol{I}^R(i,j)]^2, \tag{1}$$

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 in finding the stereo correspondence. However, there are still ambiguous stereo correspondences after using the epipolar-line constraint. This paper uses a new constraint of *mutually-supported consistency* to eliminate the unreliable stereo correspondence. The definition of this constraint is given in **Step 3** of **Algorithm 2**. An illustration of the mutually-supported consistency constraint is shown in Figure 3. The algorithm for initial stereo correspondence is described below.

Algorithm 2. Initial Stereo Correspondence: Given two sets of salient features, in the left and right images, at time $t = 0$, this algorithm determines the initial stereo correspondence pairs.

Step 1: For each distinguished feature in the left image:

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

Step 1.2: For each salient feature (distinguished or suppressed) falling in the search area, compute the similarity measure $\boldsymbol{J}$.

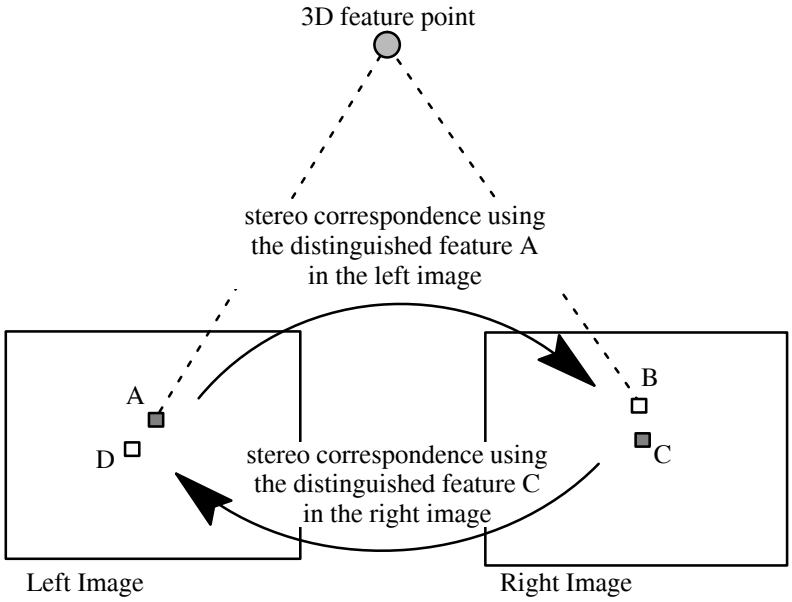

Figure 3. Illustration of the mutually-supported consistency constraint.
- distinguished features
- salient (distinguished or suppressed) features

Step 1.3: The feature having the largest ***J*** is regarded as the *possible* corresponding feature in the right image.

Step 2: For each distinguished feature in the right image,

Repeat the procedure similar to **Step 1**.

Step 3: Use the mutually-supported consistency constraint, as described below, to eliminate the unreliable stereo correspondence found in **Step 1**:

For each stereo correspondence pair (***A***, ***B***) obtained in **Step 1.**

IF there exists a stereo correspondence pair (***C***, ***D***) obtained in **Step 2**, such that

*Distance(**B**,**C**) < Consistency_tolerance*, and

*Distance(**A**,**D**) < Consistency_tolerance,*

THEN

*(**A**, **B**)* is a mutually-supported stereo pair;

ELSE

*(**A**, **B**)* is an unreliable stereo pair.

END of Initial Stereo Correspondence

III.2 Temporal Correspondence

Consider Figure 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 Figure 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 the possible temporal correspondences $\{q^L(t,k), k = 1,2,3\}$ (or $\{q^R(t,k), k = 1,2,3\}$) using the template matching method. The algorithm for 2D temporal matching is described in the following.

Algorithm 3. 2D Temporal Matching: Given a set of 2D features $\{q'^L(t)\}$ (or $\{q'^R(t)\}$) obtained from **Step 2.1** in the **Main Algorithm**, this algorithm generates a set of candidates of temporal correspondence $\{q^L(t,k), \quad k = 1,2,3\}$ (or $\{q^R(t,k), \quad k = 1,2,3\}$) 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 $q'^L(t)$, find the best temporal match by 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 best three temporal matches to be the candidates of the image location corresponding to the 3D feature being tracking.

END of 2D Temporal Matching

To reduce the computation cost, the size of the search region used in the 2D template 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.

III.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 then use the stereo correspondence module to resolve the unique stereo pair from the multiple temporal candidates in both images. As in the initial stereo correspondence, both the epipolar line constraint and the mutually-supported consistency constraint are utilized here. The details of the algorithm are shown below.

Algorithm 4: Stereo Correspondence and Triangulation: For each 3D feature in track, this algorithm resolves to find a unique stereo pair from multiple candidates of temporal correspondence, in both images, obtained from 2D temporal matching module, and then computes the estimate of the 3D feature location by stereo triangulation.
For each 3D feature in track:

Step 1: For each temporal matching candidate $q^L(t,k), k = 1,2,3$, 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 $q^R(t,k)$, find the best stereo match by computing the similarity measure $\boldsymbol{J}$ with the intensity pattern centered at $q^L(t,k)$.

Step 2: For each temporal matching candidate $q^R(t,k), k = 1,2,3$, in the right image,

repeat the procedure similar to **Step 1**.

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

Step 3.1: **IF** no mutually-supported consistency pair, **THEN**

fail to track this 3D feature for this time instant**.**

Step 3.2: **IF** only one mutually-supported consistency pair, **THEN** choose this pair to be 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 the estimate of the 3D feature location $\hat{p}(t)$ by stereo triangulation.

END of Stereo Correspondence and Triangulation

IV. CLUSTERING OF MOVING OBJECTS

To track multiple rigid objects, our system partitions the 3D feature points into several clusters, each having a common 3D motion. Once the 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 the modeling error and the measurement noise, the 3D prediction may not be good enough, and the temporal and stereo correspondences for some 3D features may occasionally go wrong and jeopardize the performance of the tracking system. In this section, we propose a new method for verifying the 3D feature correspondences over time and for clustering the 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 going through **Step 2.1 – Step 2.5** of the **Main Algorithm** 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:

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

In the following text, we may use S to represent $S(t-1,t)$ for simplicity when it causes no confusion. Our goal is to partition the 3D feature points into common-motion clusters by using the information contained in $S(t-1,t), t = 1,2,3,\cdots$. Simple clustering methods, such as the 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 problem, 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, by using the Arun method [4]. The strategy we used for clustering 3D feature points in the initial clustering stage (*t=1*) is different from that 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**. 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 with our cluster maintenance algorithm. The algorithms for generating the rigid body consensus sets are described below.

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 by using **Algorithm 6**. Those 3D features classified to belong 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 if it can be assigned as a *candidate* of a common-motion cluster in the following way:

The ith feature point, with 3D correspondence $\left(\hat{p}_i(0),\ \hat{p}_i(1)\right)$, is classified into cluster j if it gives the smallest $e_i^j = \left\| \hat{p}_i(1) - {}^1R_0^j\ \hat{p}_i(0) - {}^1T_0^j \right\|$ for all j and $e_i^j \leq$ *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 ${}^tT_{t-1}$ by using **Algorithm 7**.

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

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 having its 3D correspondence pair 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. Notice that the size of the largest rigid body consensus set generated by this algorithm can be smaller than three due the three threshold of Tolerance_Tight.

Step 1: Determine the number of random trials, $N_{trial} = \frac{k}{w^3}$ (refer to [13]).

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 ${}^{t}T_{t-1}$ by using the Arun method [4].

Step 2.2: Form the consensus set $S1^{*}$ from S, where $S1^{*}$ is the set of all $\left(\hat{p}_i(t-1),\ \hat{p}_i(t)\right) \in S$ 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 by using a weighted version of the Arun method.

END of Finding the Largest Rigid Body Consensus Set

After executing the initial clustering algorithm, each 3D feature point will be either a *member* or a *candidate* of a common-motion cluster, or still remain to be 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 $\left(\hat{p}_i(t-1),\ \hat{p}_i(t)\right)$ 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 $\left(\hat{p}_i(t-1),\ \hat{p}_i(t)\right)$ 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

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

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 ${}^{t}R^{j}_{t-1}$ and ${}^{t}T^{j}_{t-1}$, 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 $\left(\hat{p}_i(t-1),\ \hat{p}_i(t)\right)$, we define its *motion similarity error* with respect to cluster j having motion ${}^{t}R^{j}_{t-1}$ and ${}^{t}T^{j}_{t-1}$ to be

$$e^{j}_{i} \equiv \left\| \hat{p}_i(t) - {}^{t}R^{j}_{t-1}\ \hat{p}_i(t-1) - {}^{t}T^{j}_{t-1} \right\| . \tag{3}$$

If the motion similarity error of a 3D feature P_i with respect to cluster j, e^{j}_{i}, is the smallest among all 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. Also, two common-motion clusters, j and k, are said to have common 3D motion if the following two conditions are both satisfied:

$$\frac{1}{\#\left({S_M}^{j}\right)} \sum_{(\hat{p}(t-1),\,\hat{p}(t))\in {S_M}^{j}} \left\| \hat{p}(t) - {}^{t}R^{k}_{t-1}\ \hat{p}(t-1) - {}^{t}T^{k}_{t-1} \right\| < Tolerance\_Tight\ , \tag{4}$$

$$\frac{1}{\#\left({S_M}^{k}\right)} \sum_{(\hat{p}(t-1),\,\hat{p}(t))\in {S_M}^{k}} \left\| \hat{p}(t) - {}^{t}R^{j}_{t-1}\ \hat{p}(t-1) - {}^{t}T^{j}_{t-1} \right\| < Tolerance\_Tight\ . \tag{5}$$

The algorithm for maintaining the common motion clusters is described below.

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 ${}^{t}R^{j}_{t-1}$ and ${}^{t}T^{j}_{t-1}$ by using **Algorithm 7**.

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

THEN

Assign all its members and candidates to be 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 ${S_M}^j(t-1,t) \setminus A$ from ${S_M}^j(t-1,t)$ to the un-clustered set S^0. That is, the inconsistent *members* are removed from the *j*th cluster and assigned to be un-clustered.

Step 1.2.2: For each candidate in the *j*th common-motion cluster, check if it can be promoted as a *member* in the following way:

If its motion similarity error with respect to the *j*th cluster is smaller than *Tolerance_Tight*, then promote it to be a member; otherwise, remove it from the candidate set of the *j*th cluster and assign it to be an un-clustered feature, for the moment, which 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 if it can be assigned 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 assign it to be a *candidate* of that most similar-in-motion cluster; otherwise, keep it to be un-clustered.

Step 3: If $\#\left(S^0\right)$ 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 common 3D motion 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 assign all its members and candidates to be un-clustered and delete this cluster (i.e., this common-motion cluster disappears).

End of Cluster Maintenance

Notice that in **Algorithm 8**, **Step 1.2** and **Step 4.2** deal with the disappearance of an active cluster; **Step 3** accomplishes the appearance of new clusters; **Step 4.1** deals with the merging of clusters; and **Step 1**, together with **Step 3**, accomplishes the splitting of an active cluster. In our algorithm, only members contribute to the motion estimation of a common-motion cluster; candidates are only assigned to a common-motion cluster for predicting their next 3D position and for the possibility of becoming a member of that cluster in the next time instant. Consider Figures 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 leave as 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 be 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 be assigned as a candidate of a cluster (**Step 2**), or remain as an un-clustered feature (**Step 2**). Before becoming a member of a common-motion cluster, a 3D

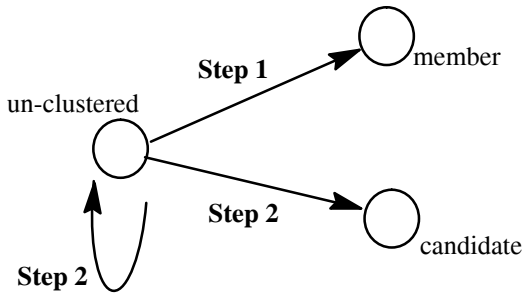

Figure 4. State transition of a 3D feature in the initial clustering stage (**Algorithm 5**).

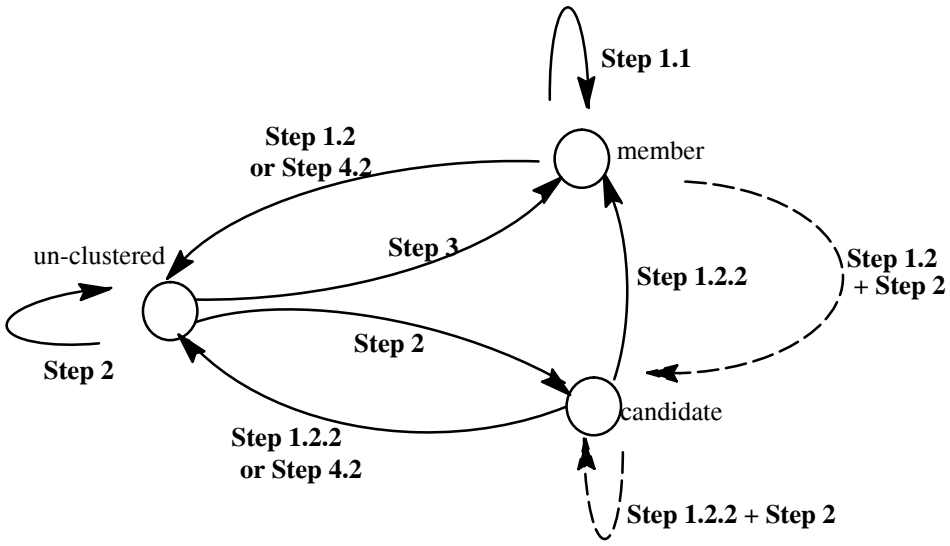

Figure 5. State transition of a 3D feature in one iteration of the clustering maintenance stage (**Algorithm 8**).

feature has to first become a candidate of that cluster, unless it is one of the initiator of that cluster. In one iteration of the clustering maintenance algorithm, a member can become a candidate only through 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 more similar-in-motion cluster than the present one. Similarly, a candidate of a common-motion cluster can remain as a candidate only through **Step 1.2.2** and **Step 2**, i.e., it has to be first assigned as un-clustered in **Step 1.2.2**, and then become a candidate again in **Step 2**. This also gives it a chance of becoming 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 Kalman filter 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 performance on track-

ing can be achieved. To describe the Kalman filters for motion prediction, we need to address some of the modeling issues on motion kinematics.

V.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

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

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.

A common approach to modeling the motion kinematics is to divide the 3D motion of a rigid body into two parts: a rotation around an axis (called the rotation axis) and a translation of the rotation axis[27]. Let Q_0 be a fixed 3D point on the rotation 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

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

From equations (6) and (7), we have

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

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

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

where $v(t-1)$ and $a(t-1)$ are the translational velocity and acceleration , respectively. 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,

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

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 $\mu(t) = \mu(t-1)$.

V.2 3D Feature Prediction by Using Kalman Filters

In our system, Kalman filters are used to predict 3D motions of multiple rigid objects. The predicted 3D motions are then used to predict the 3D feature locations for simplifying 2D temporal matching (or "tracking") in the next time instant. Linear Kalman filters, 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

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

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 equations (9) and (10), the state equation can be written as

$$s_{t+1} = H\ s_t + n_t\ , \tag{12}$$

where the state transition matrix H is

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

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

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

An important step in applying *linear* Kalman filtering to the motion prediction problem is to use the motion estimates obtained by the motion clustering and estimation module, instead of directly using the 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 equations (7) and (9), we have

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

by 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 ${}^{t}R_{t-1}$ by using the Rodrigues formula [15]. Define the measurement vector to be

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

By using equation (15), the measurement equation can be written as

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

where

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

and the measurement noise η_t is white with covariance matrix Λ_t, i.e.,

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

Figure 6 shows the block diagram of the module for 3D feature prediction by Kalman filtering and its interface with the rest part of the system (also refer to Figure 3). 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 part 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 ${}^{t}T_{t-1}$, for each "rigid" body ("rigid" from the viewpoint of the observer based on recent observation). The Kalman filter can be divided into two parts: updating and prediction. Before the updating, the Kalman gain matrix should be estimated by following equation:

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

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

$$P_{t|t-1} = H_{t-1} P_{t-1} H_{t-1}^T + Q_{t-1} \ . \tag{21}$$

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

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

and the state covariance matrix P_t is updated by:

$$P_t = (I - K_t F_t) P_{t|t-1} \ . \tag{23}$$

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

$$\hat{s}_{t+1|t} = H \hat{s}_t \ . \tag{24}$$

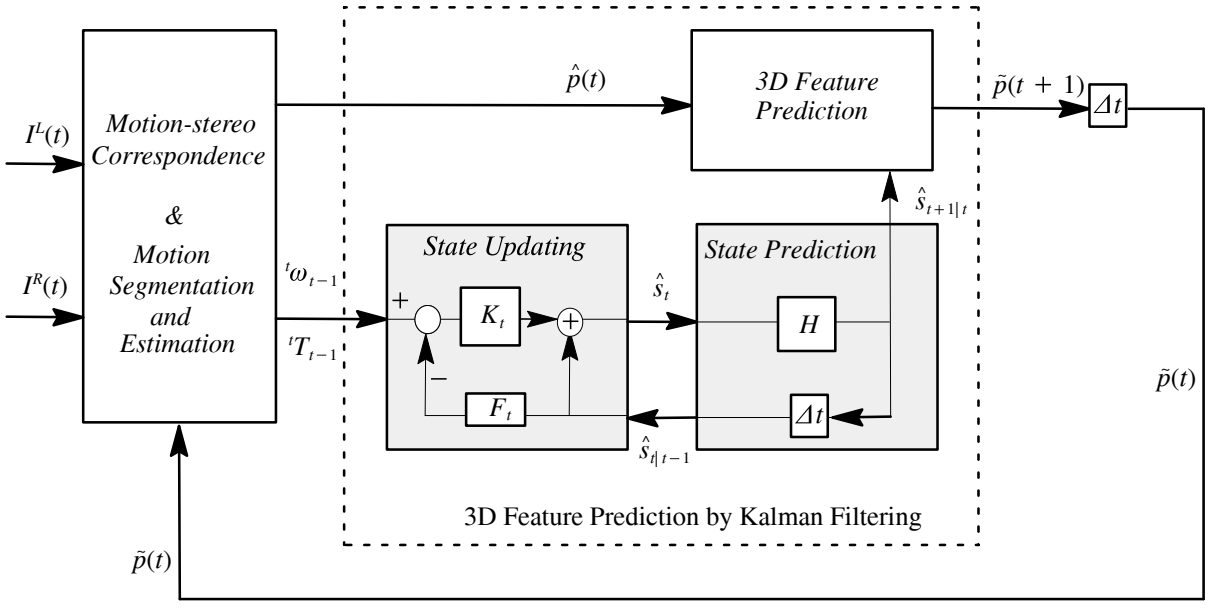

Fig 6. The block diagram of the module for 3D Feature Prediction by Kalman Filtering and its interface with the rest part of the system shown in Figure 3.

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

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

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 $\hat{s}_{t+1|t}$) by using the Rodrigues formula [15].

VI. EXPERIMENTAL RESULTS

Fig. 7 shows the active binocular head used in the experiments described below. The camera parameters and the kinematic parameters of the binocular head were calibrated in advance [22][23], and could be controlled to fixate on any given 3D points. For our stereo vision system, the error for 3D point measurements, due to calibration inaccuracy and 2D feature detection error, is less than 2 millimeters in general. Therefore, *Tolerance_Tight* is set to 2 millimeters, which is the uncertainty of the 3D point measurement. On the other hand, *Tolerance_Loose* is arbitrarily set to 5 millimeters

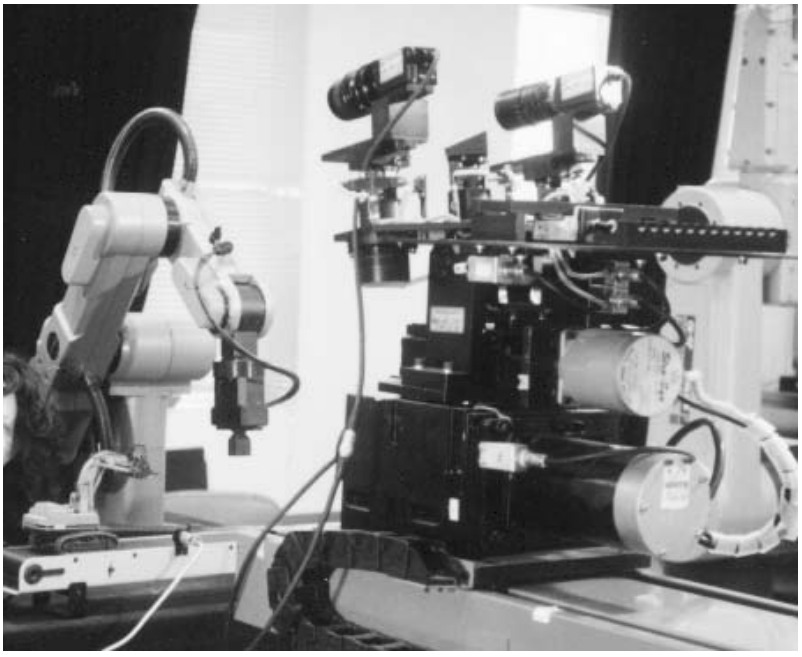

Fig. 7. The active binocular head used in our experiment.

in order to allow some non-rigidity or slight mis-correspondence. $N_{model\_required}$ is three because the Arun method requires at least three point correspondences to compute the motion parameters, and $N_{\min}$ is arbitrarily set to ten because a rigid body should have at least ten features in our experiments. In the following experiments, the window size for similarity test used in the template matching module is 7×7 pixels, and the *consistency_tolerance* used for checking the mutually-supported consistency was set to 2 pixels. Because of the sensitivity problem for estimating angular acceleration, we assume constant angular velocity in our system, i.e., the 3D object undergoes a *constant* 3D rotation *during a time interval* and $\omega(t) = \omega(t-1)$.

This section presents the results of the tracking experiments on three motion sequences, each of them having 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 right to left as in the first image sequence, and the binocular head was panning (0.2° per frame) from right to left while the background was stationary. In the last 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 moved approximately 7 millimeters per frame. Figures 8, 10 and 12 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 5x5 squares. The initial stereo correspondences obtained with our automatic matching algorithm are quite reliable, which gives a good foundation for the subsequent tracking. There are 56~73 features on the cola can and 136~144 features on the background, depending on the image sequence, the time instant, and the left or right image.

Figures 9, 11 and 13 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 were marked with small crosses 5 pixels high. Square marks indicate the 2D locations, in the last image, of those

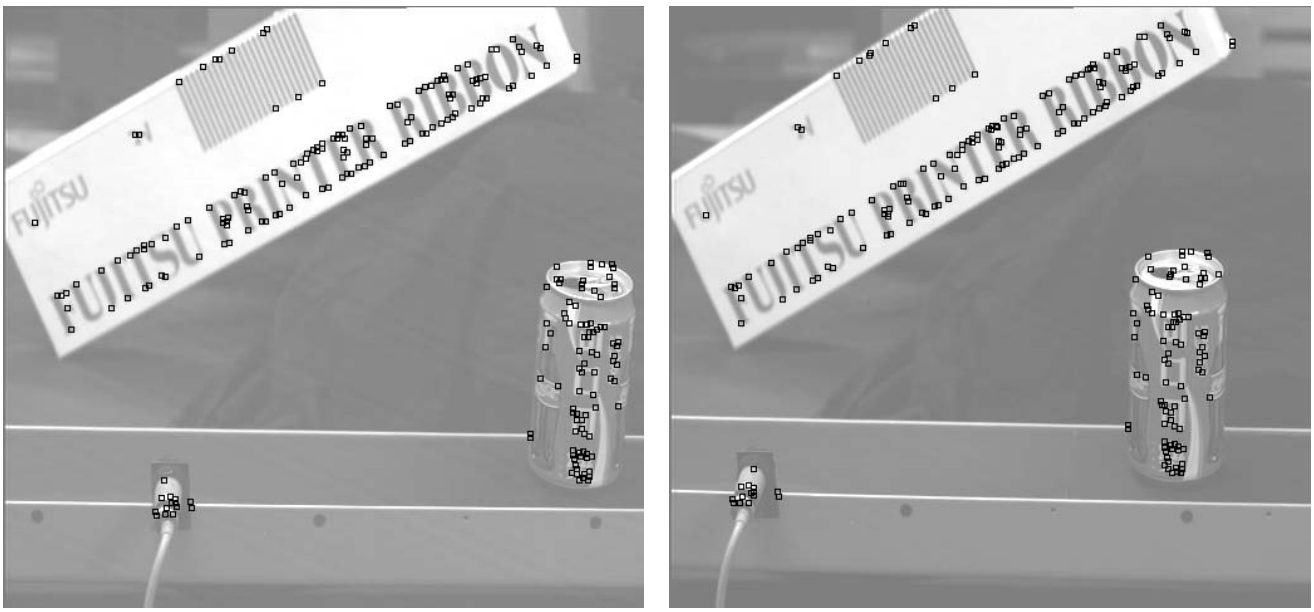

Figure 8. (Image sequence 1) Initial stereo correspondence pairs superimposed on the first left and right images.

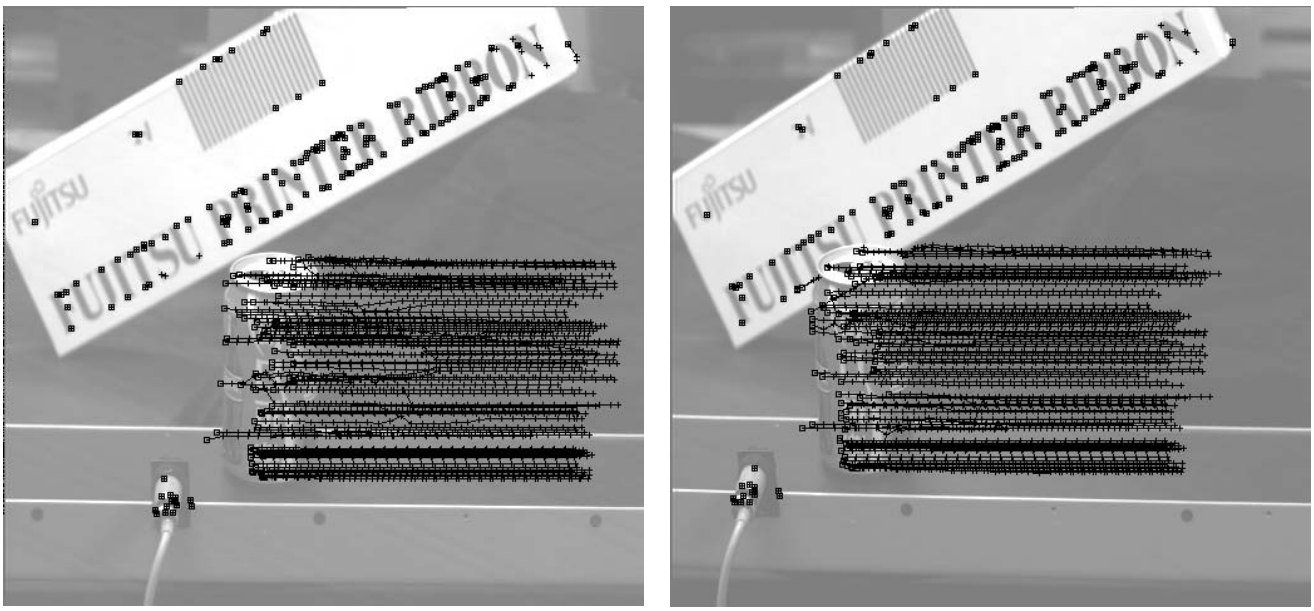

Figure 9. (Image sequence 1) Trajectories of the tracked feature points superimposed on the last stereo image pair.

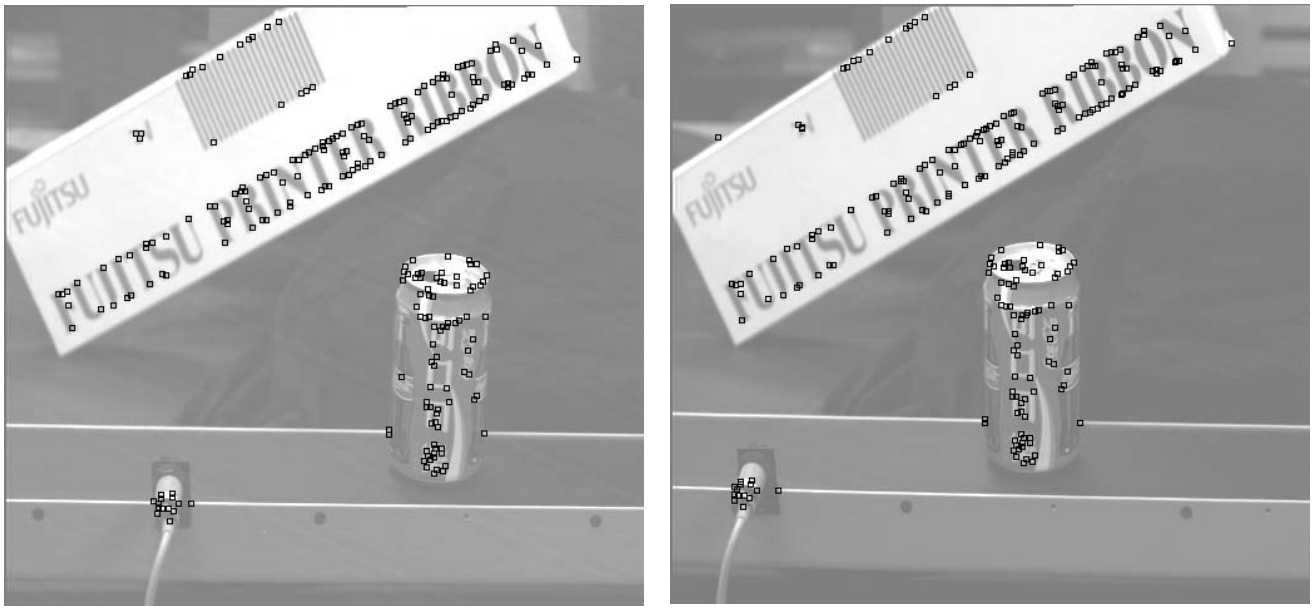

Figure 10. (Image sequence 2) Initial stereo correspondence pairs superimposed on the first left and right images.

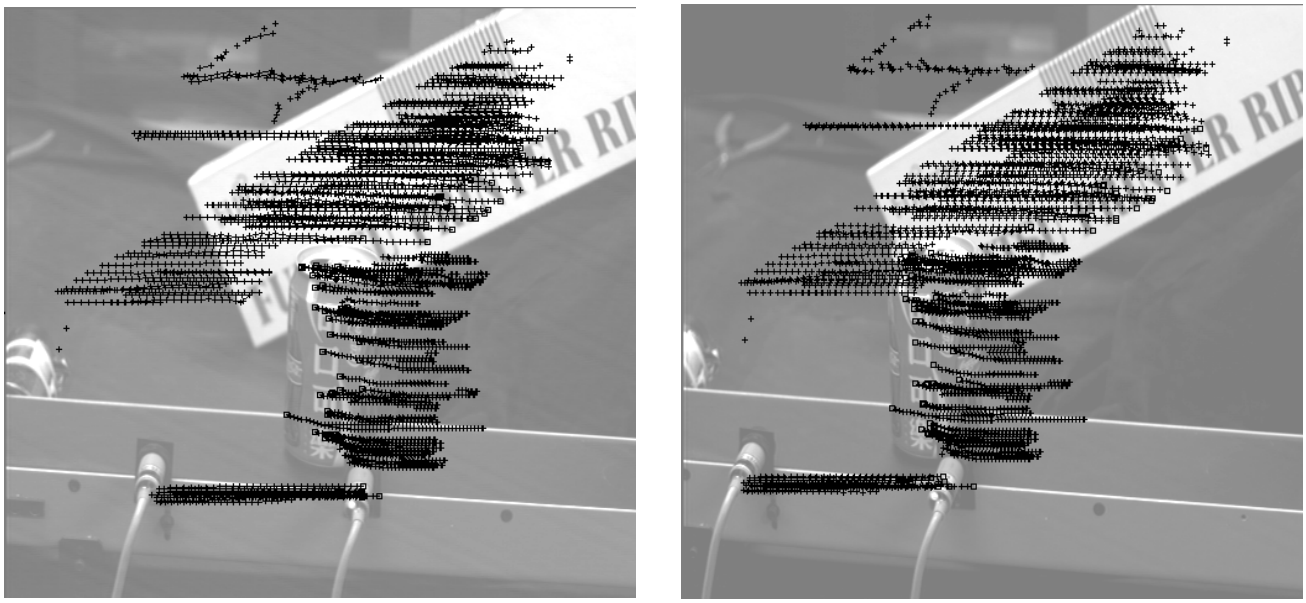

Figure 11. (Image sequence 2) Trajectories of the tracked feature points superimposed on the last stereo image pair.

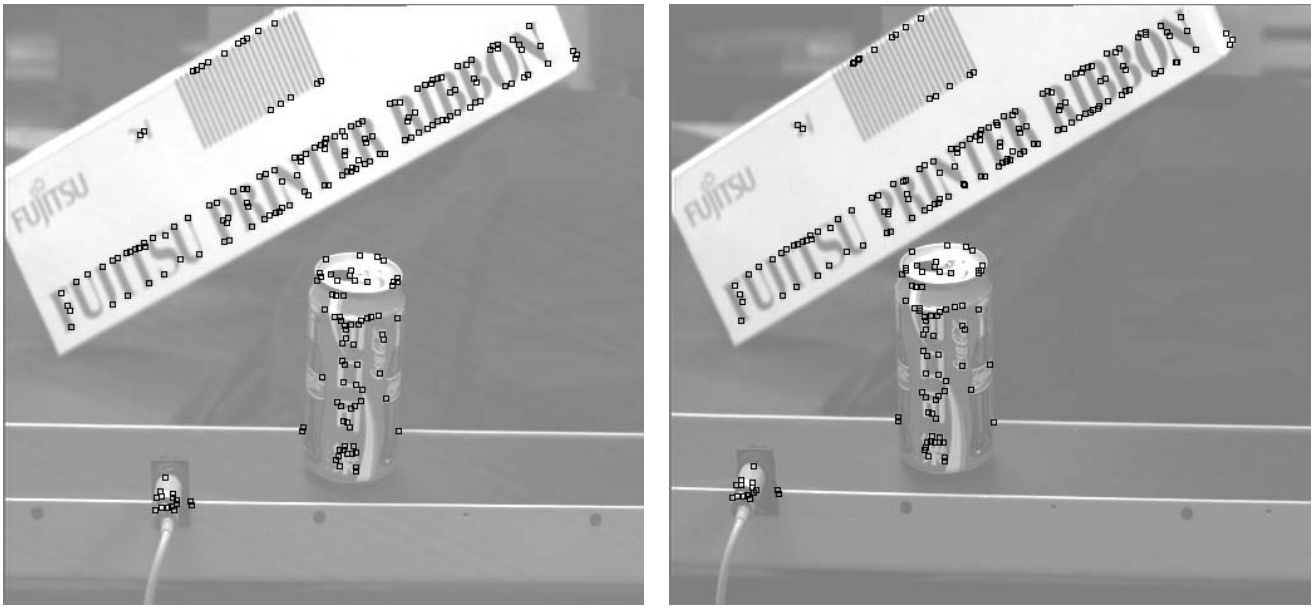

Figure 12. (Image sequence 3) Initial stereo correspondence pairs superimposed on the first left and right images.

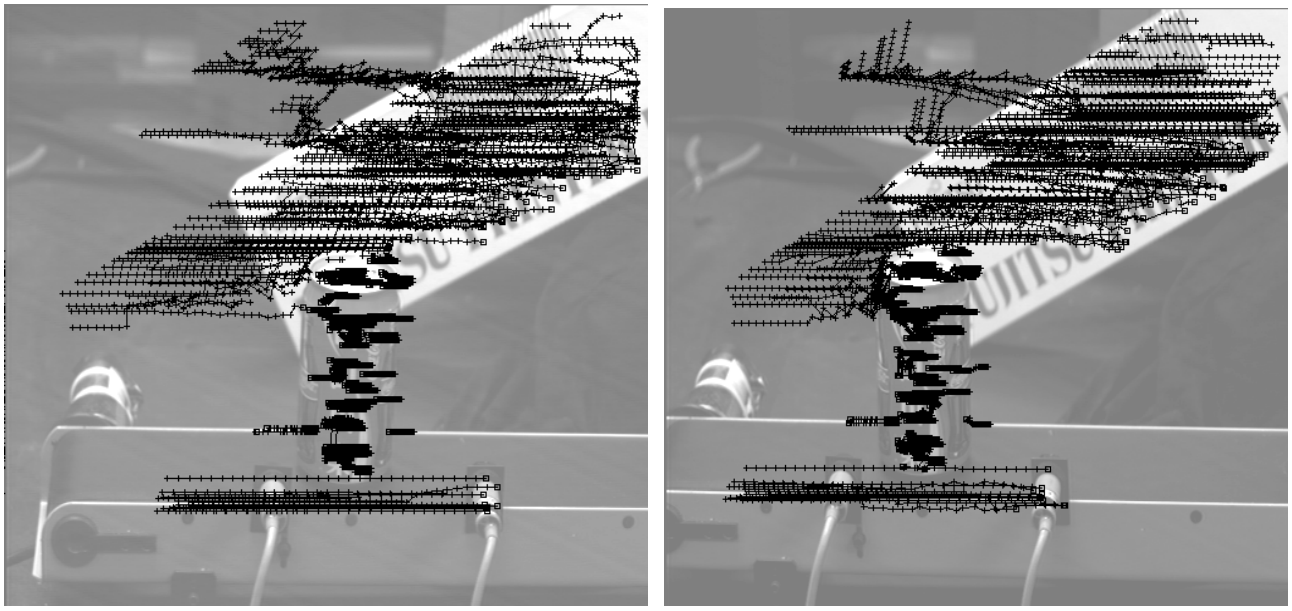

Figure 13. (Image sequence 3) Trajectories of the tracked feature points superimposed on the last stereo image pair.

feature points which were still in track after 30 image frames. Objects having different 3D motions were successfully clustered and segmented by 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 an interesting object by applying gage control so that the object remains in the field of view. 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 the cameras around) to fixate its cameras on the object it is interested in. This tracker is completely autonomous in the sense that it requires no initial correspondence of any kinds, either temporal or stereo correspondence. Any 3D rigid objects, or any articulated objects such as robot manipulators, can be tracked with our tracker as long as there are some corner features on each moving component. We are currently extending it to also utilizing 3D line features. We are also working on extending this system to track slightly non-rigid object by adaptively adjusting the tolerance for testing motion similarity and by considering the spatial relationship between the 3D features. If at some stage, the object is recognized (or hypothesized) to be a known object with some parametric model, then techniques used by Koller, Daniilidis and Nagel [16] can be applied in the 2D template matching module to enhance the performance of our tracker.

The performance of our 3D tracker is mainly based on the following factors: (i) A RANSAC-based motion clustering and estimation algorithm using the rigid body consensus is proposed for grouping features into common-motion clusters and estimating their 3D motion simultaneously. This clustering method provides a systematic way for managing splitting, merging, new appearance and disappearance of multiple moving rigid objects. (ii) Linear Kalman filters are used to predict the next movements of common-motion clusters, which can then be used to provide better prediction for tracking 3D features (i.e., finding temporal correspondence). (iii) Two parallel 2D temporal modules are utilized to make full use of the temporal information contained in the image sequence. (iv)

The constraint of mutually-supported consistency is introduced to eliminate incorrect stereo correspondences. (v) Calibration is done automatically and accurately to assure the accuracy of 3D inference. (vi) In order to exploit parallelism to meet the real-time requirement, our system is integrated with multi-agent architecture, which also makes it easiest to extend our system by adding extra agents. 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.

ACKNOWLEDGEMENTS

We would like to thank Dr. Leslie Kitchen for his help in the implementation of the feature extraction module and some initial discussions on system design during his sabbatical stay with us. We also want to thank Dr. Lin-Guo Liou for some useful suggestions on motion modeling and tracking. This work was supported in part by the National Science Council, Taiwan, under grants NSC 83–0408–E–001–010 and NSC 84–0408–E–001–004.

REFERENCES

[1] N. Ahuja and A. L. Abbott, "Active Stereo: Integrating Disparity, Vergence, Focus, Aperture, and Calibration for Surface Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 15, No. 10, Oct. 1993, pp. 1007–1029.

[2] P. K. Allen, A. Timcenko, B. Yoshimi, P. Michelman. Automated Tracking and Grasping of a moving Object with a Robotic Hand-Eye System, *IEEE Trans. on Robotics and Automation*, Vol. 9, No. 2, Apr. 1993, pp. 152–165.

[3] J. Y. Aloimonos, I. Weiss and A. Bandopadhay, "Active Vision", *IEEE Proceedings of the 1st Int. Conf. on Computer Vision*, Jun. 1987, pp. 35–54.

[4] K. S. Arun, T. S. Huang, and S.D. Blostein, "Least–Square Fitting of Two 3–D Point Sets," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 9, NO. 5, pp. 698–700, 1987.

[5] R. Bajcsy, "Active Perception", *Proc. IEEE* **76**(8), 1988, pp. 996–1005.

[6] T. K. Cheng, L. Kitchen. Multi–Agent 3D Tracking and Segmentation Using Stereo, Department of Computer Science, University of Melbourne. Technical Report 1993/30.

[7] H. Christensen, "A Low–Cost Robot Camera Head", *Int. J. Pattern Recogn. Artif. Intell.* Vol. 7, No. 1, 1993, pp. 69–87.

[8] J. Cooper, S. Venkatesh and L. Kitchen, "Early Jump–Out Corner Detectors," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 8, 1993, pp. 823–828.

[9] J. Cooper, "Real–Time Task–Directed Robot Vision," Ph. D. thesis, The Department of Computer Science, University of Western Australia, 1994.

[10] J. Crowley, P. Bobet and M. Mesrabi, "Gaze Control for a Binocular Camera Head", *Proceedings SPIE*, Application of AI X: Machine Vision and Robotics, Orlando, Apr. 1992, pp. 47–61.

[11] N. J. Ferrier and J. J. Clark, "The Harvard Binocular Head," *Int. J. Pattern Recogn. Artif. Intell.* Vol. 7, No. 1, 1993, pp. 9–31.

[12] J. C. Fiala, R. Lumia, K. J. Roberts, and A. J. Wavering, "Triclops: A Tool for Studying Active Vision", *Int. J. Computer Vision*, Vol. 12, No. 2/3, 1994, pp.231–250.

[13] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, Jun. 1981, pp. 381–395.

[14] M. S. Lew, T. S. Huang, and K. Wong, "Learning and Feature Selection in Stereo Matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 9, Sep. 1994, pp. 869–881.

[15] K. Kanatani, *Group-Theoretical Methods in Image Understanding*. Springer-Verlag 1990.

[16] D. Koller, K. Daniilidis and H. H. Nagel, "Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes," *Int. J. Computer Vision*, Vol. 10, No. 3, 1993, pp.257–281.

[17] E. P. Krotkov and R. Bajcsy, "Active Vision for Reliable Ranging: Cooperative Focus, Stereo and Vergence," *Int. J. Computer Vision*, Vol. 11, No. 2, 1993, pp. 187–203.

[18] E. Milios, M. Jenkin, and J. Tsotsos, "TRISH: A Binocular Robot Head with Torsional Eye Movements", *Proceedings SPIE*, Application of AI X: Machine Vision and Robotics, Orlando, Apr. 1992, pp. 36–46.

[19] K. Pahlavan and J. O. Eklundh, "A Hand-Eye System-Analysis and Design," *Comput. Vision Graph. Image Process. : Image Understanding* 56, 1(1992), pp. 41–56.

[20] N. P. Papanikolopoulos, "Controlled Active Vision," Ph. D. thesis, The Department of Electrical and Computer Engineering, Carnegie Mellon University, 1992.

[21] N. P. Papanikolopoulos, B. Nelson and P. K. Khosla, "Monocular 3-D Tracking of a Moving Target by Eye-In -Hand Robotic System," *Proceedings of the 31th Conf. on Decision and Control*, Tucson, Arizona, Dec. 1992, pp. 3805–3810.

[22] S. W. Shih, Y. P. Hung and W. S. Lin. "Accurate Linear Techniques for Camera Calibration Considering Lens Distortion by Solving an Eigenvalue Problem," *Optical Engineering*, vol. 32, No. 1, 1993, pp. 138–149

[23] S. W. Shih, Y. P. Hung and W. S. Lin, "Kinematic Parameter Identification of a Binocular Head Using Stereo Measurements of a Single Calibration Point," to appear in Proceedings IEEE International Conference on Robotics and Automation, May 1995.

[24] J. Weng, T. Huang and N. Ahuja, "3-D Motion Estimation, Understanding, and Prediction from Noisy Image Sequences," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 9, No.3, 1987, pp. 370–389.

[25] C. H. Yang, J. J. Leou, "Robot Tracking by Stereo Vision," *Proceedings of 1993 Conf. on Computer Vision, Graph. and Image Processing*, pp. 255–262.

[26] G. S. Young and R. Chellappa, "3–D Motion Estimation Using a Sequence of Noisy Stereo Images: Models, Estimation, and Uniqueness Results," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, No. 8, 1990, pp. 735–759.

[27] Z. Zhang and O. Faugeras, *3D Dynamic Scene Analysis: A Stereo Based Approach*. Springer-Verlag 1992.

[28] Z. Zhang and O. Faugeras, "Three–Dimensional Motion Computation and Object Segmentation in a Long Sequence of Stereo Frames," *Int. J. Comput. Vision* Vol. 7, No. 3, Mar. 1992, pp. 211–241.