Internal report number: khw_irep_070516

The CSE Dept., The Chinese University of Hong Kong, Internal report, 2007-May-16

Extended Kalman Filter Based Pose Estimation Using Multiple Cameras

M.E. Ragab, and K.H. Wong

Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong.
E-mail: khwong@cse.cuhk.edu.hk

Abstract- In this work, we solve the pose estimation problem for robot motion by placing multiple cameras on the robot. In particular, we combine the Extended Kalman Filter (EKF) with the multiple cameras. An essential strength of our approach is that it does not require finding image feature correspondences among cameras which is a difficult classical problem. The initial pose, the tracked features, and their corresponding 3D reconstruction are fed to the multiple-camera EKF which estimates the real-time pose. The reason for using multiple cameras is that the pose estimation problem is more constrained for multiple cameras than for a single camera, which has been verified by simulations and real experiments alike. Different approaches using single and two cameras have been compared, as well as two different triangulation methods for the 3D reconstruction. Both the simulations and the real experiments show that our approach is fast, robust and accurate.

1. Introduction

To find the pose of an object is to get its position and orientation. It is a popular research problem, and is related to many different areas such as: robotics, man-machine interaction, augmented reality (AR), and intelligent vehicle guiding [37]. Applications are abundant, for example, maintenance training by augmented reality [23], precise localization in industrial environments [39], and identifying large 3D objects [27].

The pose estimation problem can be solved by many different approaches such as: dead reckoning, range sensing and map fusion [9]. However, most of these approaches suffer from drawbacks, e.g. the increase of systematic errors with time [37]. On the other hand, the computer vision approaches can solve this problem more efficiently and with lower cost, especially with the dropping prices and the increasing accuracy of cameras.

Pose estimation has different flavors in the field of computer vision. Sometimes it is denoted as the 2D-3D absolute orientation problem [13] which is equivalent to getting the (ego-) motion of the camera. If the 3D structure of the scene is required, then the problem becomes Structure from Motion (SfM). If it is required to find the relative camera pose between two views, the problem is denoted as the 2D-2D relative pose problem. When it is required to register a 3D object to a 3D CAD model, the problem is denoted as the 3D-3D pose problem [2], and [40]. There is an ambiguity problem in pose estimation which is known as the bas-relief ambiguity [3], [31], and [35]. This ambiguity becomes obvious when the depth variation of the scene is small, and the camera field of view is narrow leading to misinterpreting the small translation along one axis as a rotation around another and vice versa.

In this work, we are interested in finding the ego-motion of a set of cameras for the sake of localizing a robot moving within a scene, however we intend to extend in future work to obtain an accurate 3D reconstruction of the scene on the fly. One approach of the solution is the use of epipolar geometry [14] based on the fundamental matrix or trifocal tensors. This approach is linear however it suffers from degenerate configurations and requires time consuming statistical techniques such as RANSAC to reduce the ruining effect of outliers.

Another approach to find the camera motion is the bundle adjustment (BA) [14] and [33], which finds the optimal solution; however it requires a good initialization and a high computational cost. Factorization [14], [26], and [32] is optimal for affinities. Yet, the limiting factor of both BA and factorization is that they are batch techniques, which are not suitable for real time applications such as the problem at hand.

When it is required to solve the problem in real time, as in our case, we have to use recursive techniques such as the Kalman filter (KF) and its variants or the particle filter/ CONDENSATION algorithm. The latter, the particle filter, is more advantageous in tracking continuous curves, such as heads or hand contours, in dense visual clutter. However, KF variants are quite satisfactory in computational speed for point targets, as in our case where feature corners are tracked among frames. Moreover, improving the performance of the CONDENSATION algorithm requires increasing the sample size and the computational cost [16].

KF [15], [16], and [22] is an optimal recursive data processing algorithm for linear systems [19]. However, the linear assumption is violated by the perspective camera model. In this case, using Extended Kalman Filter (EKF) is necessary. In fact EKF has been used in diverse ways in the field of computer vision [7], [4], and [17]. The work in [8] by Broida et al was one of the early attempts to bring Kalman filter to the field of computer vision and motivated the related researches that followed. Broida et al used one filter for the pose and 3D structure, and used iterated extended Kalman filter (IEKF) to encounter the high nonlinearity of the state-measurement relation and the divergence of EKF in some cases. Weng et al in [36] compared batch to recursive techniques in motion and structure estimation. They also used one filter for both pose and structure and IEKF to improve the performance. Azarbayejani et al in [1] enhanced the recursive technique further to obtain the focal length which is one of the camera intrinsic parameters, and indicated the effect of parameterization on the stability and accuracy of the filter. They used one filter for both pose and structure and iterated EKF once to remove the initial transient. In [11], one EKF for both pose and structure is used. Having one filter for both pose and structure guarantees that they are coupled, however in this way, the length of the state space vector becomes large which may affect the filter stability. It is noted in [1] that the larger the length of the state vector, the lower the stability of the filter. Therefore, in this case, tuning the filter would be rather difficult.

This may be the stimulus behind using separate filters for pose and structure. In [6], one separate IEKF filter for each 3D point was used for structure updating. The pose was obtained using epipolar geometry and the accuracy was improved using RANSAC approach which rendered the solution not suitable for real time implementations as indicated above. In [38] Yu et al. used one EKF filter for pose as the first step, then used a set of EKFs, one for each model point as the second step. This in fact improves speed but may lower the accuracy due to the decoupling of pose and structure. In [29], Shademan et al. carried out a sensitivity analysis of EKF and IEKF in pose estimation but using a few number of points (about 5 or 6 points).

Besides using multiple cameras in stereo rigs, they have been used in pose estimation primarily to resolve the bas-relief ambiguity [3]. In [10], [12], and [25], the multiple cameras are dealt with as a single generalized camera then, in [10], the problem is formulated in a least–squares manner and solved iteratively. Sometimes multiple cameras are used with KF or EKF, for example, in [28] multiple fixed cameras, and a KF combined with eigenspace methods are used to observe the pose of a planar robot joint. Additionally Lippiello et al in [18] discussed using multiple fixed cameras for pose estimation of an object with a known CAD model.

In contrast, we use two cameras put back-to-back on a robot moving within the scene. The inputs to the system are the simultaneous frames taken by each camera, camera intrinsic parameters, and the initial pose. The output is the real time pose for each frame along a sequence of one hundred frames. To do this job, we use one EKF for pose estimation, and though our main interest is recovering pose, we calculate the 3D structure of the points fed to the EKF as well. To make the 3D structure consistent with the pose, we calculate it using triangulation methods based on the pose obtained from the EKF as will be explained in section 3. Furthermore, feature correspondences are not sought between different cameras in the set, only in subsequent frames of the same camera which increases the accuracy of tracking due to the small baseline between frames. The fairly small number of sought correspondences for each camera drastically removes outliers. This configuration is promising in resolving the bas-relief ambiguity, and would be able to recover the pose in case of any of the cameras being off or for example imaging a textureless surface without enough features to track.

1.1 Contributions

The main contributions of our work are (1) formulating the EKF implementation for the pose estimation of moving multiple cameras, (2) comparing a triangulation method based on analytic geometry, the AG triangulation, to the well known linear triangulation method in terms of the ability to recover the scale factor, speed, and accuracy, and (3) using a changeable set of features to avoid the effect of occlusion.

1.2 Structure of the Rest of the Paper

In section 2, we present the background which discusses the scale factor ambiguity, the linear triangulation method. In section 3, we present details of our work, the layout of the solution algorithm, and the EKF implementation. In section 4, we introduce the experiments and the results. Section 5 is a discussion. We conclude our work in section 6. Appendix A is dedicated to the AG triangulation, and at last, appendix B discusses the coordinate systems to be used as a reference for the motion.

2. Background

2.1 Scale Factor, Absent and Present

The scale factor ambiguity has attracted a lot of attention in the literature. For example, in [8] it is indicated that normalizing the states is necessary to eliminate a free scale factor, otherwise, the IEKF simply diverges unpredictably along this extra degree of freedom. In [1], scale is set by setting the initial variance of the depth of the first feature point to zero. This fixes that parameter to its initial value. All other parameters automatically scale themselves to accommodate this constraint. In [11], the scale factor is associated to a three reference features which are replaced with the most suitable points when occluded leading to an inevitable drift.

Szeliski and Kang say in [30] that the scale ambiguity can be removed knowing the absolute distance between camera positions. Trucco and Verri in [34] agree with them. In [24], Nistér says that the overall scale of the configuration can never be recovered solely from images. In [20] McReynolds and Lowe comment on the viewpoint of Szeliski and Kang saying that the global depth offset cannot be determined from the images and indicate, in [20], that they use a known image scale factor which is related to the focal length and the camera's field of view. In [14] Hartely and Zisserman indicate that to know the scale we need to know the real distance between two points in the image, and even assume a unity scale factor constraint for 3D reconstructions of image sequences. In our work, we need to resolve this ambiguity as shown in section 3.

2.2 Linear Triangulation Method

This method is used for reconstructing the 3D structure knowing the camera matrices which contain the cameras' intrinsic parameters (focal length, etc.), and the extrinsic parameters (poses). This method will be explained briefly and the reader is referred to [14] for more details.

Suppose that we have 5 cameras:

$$x = PX,\ x' = P'X,\ \ldots\ldots\ldots\ldots\ldots\ldots,\ x'''' = P''''X \qquad (1)$$

Where $\boldsymbol{x} = \{x, x', \ldots, x''''\}$ are 2D image points, $\boldsymbol{P} = \{P, P', \ldots, P''''\}$ are camera matrices and X is the imaged 3D point which is found as follows:

- Compose a cross product: $\boldsymbol{x} \times \boldsymbol{P}X = 0$ to get an equation of the form $AX = 0$.
- X is the last column of V, where $A = UDV^T$ is the singular value decomposition

(SVD) of A.

Although in [14], it is mentioned that this solution is obtained under the constraint $\|X\| = 1$, we have found that this method is capable of finding the scale factor since we use the absolute camera centers in composing the camera matrices. In contrast, if the relative camera centers, obtained under the assumption of unity translations, are used, the ambiguity cannot be resolved. Another triangulation method based on analytic geometry, the AG triangulation, is described in Appendix A. Moreover, both triangulation methods are compared in subsection 5.1.

3. Our Work

3.1 Introduction

Our aim is to find the pose of a robot during its motion within the scene. Our strategy for the solution is to find the pose of a set of multiple cameras (with fixed relative positions) carried by the robot. For this implementation, we use two cameras however the algorithm is ready-to-use with any number of cameras. The reasons for using multiple cameras are: the rotation and translation are much better constrained for a multiple camera system than for single cameras [3], and [25], there are ambiguous scenes which require a set of images for accurate pose estimation [25], and intuitively adding an additional camera is equivalent to adding another eye for collecting more information. The use of multiple cameras is much more justified by the facts that the camera prices are dropping while the accuracy is increasing, and the redundancy is useful in case of any camera becomes off, completely occluded, or encountering a textureless part of the scene without enough features to track. The choice of the back-to-back setting of the two cameras is motivated by the analysis of the Fisher Information Matrix by Pless [25] however the optimal number of cameras, rotations and translations between them. The inputs to our algorithm are the image frames taken by the multiple cameras, and the camera calibrations. The output is the real time pose per frame with reference to the initial position of the first camera.

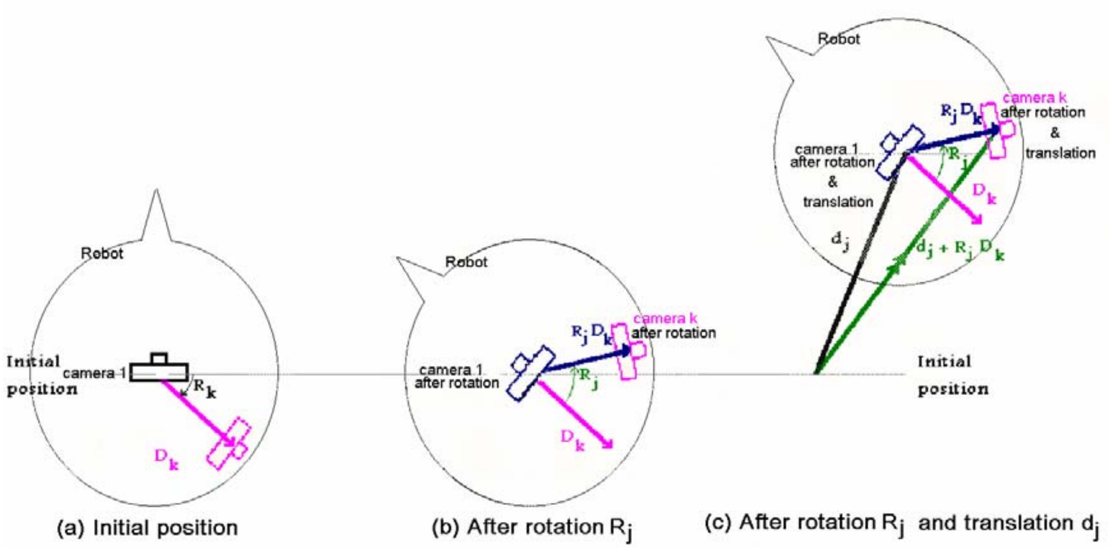

Fig.1. Top-down view: effect of rotation and translation on the displacement between cameras (referred to the initial position, R_k and D_k are fixed, and k=2 in case of using two cameras)

3.2 Multiple-Camera Model

Taking the camera at the first frame as the reference, the camera coordinates, a 3×1 vector P_{ij}, of the 3-D feature M_i (3×1 vector), at frame j is given by:

$$P_{ij} = R_j^T (M_i - d_j) \tag{2}$$

Where R_j is the (3×3) camera rotation matrix at frame j (referred to the first frame), and d_j is the (3×1) camera translation vector at frame j (referred to the first frame).

If we have multiple cameras, the k^{th} camera rotated R_k at the first frame, and translated D_k from the reference has the following j^{th} frame camera coordinates of M_i:

$$P_{ijk} = R_k^T \; R_j^T (M_i - d_j - R_j D_k) \tag{3}$$

In Fig.1, $R_{k,}$ and D_k are shown for the k^{th} camera, while R_1 and D_1 are the identity matrix and the zero vector respectively (reference position.)

Sometimes, it is required to have the reference coordinate system parallel to a fixed coordinate system in the scene, e.g. that formed by a room-corner. In most cases, it is easy to adjust the initial position of the first camera with zero angles of rotations around that fixed coordinate system however we may need some effort to do this job if the camera has a spherical shape for example. Appendix B deals with this situation.

3.3 Proposed Algorithm

We solve the pose problem sequentially, a frame by frame, throughout all the sequence (one hundred frames), which is divided into sections (ten frames each). The length of a section is a trade-off between speed and accuracy as will be shown below.

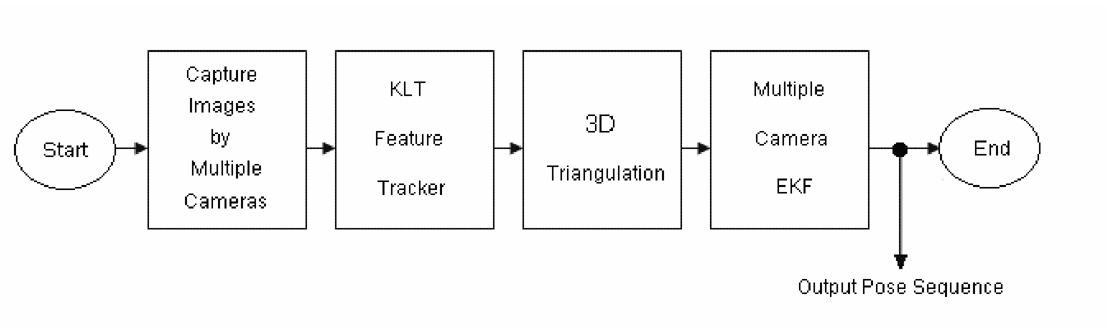

Fig.2. Brief Algorithm Flowchart.

3.3.1 Initialization

At this stage, we get the initial pose of the cameras, i.e. robot. We need only the pose of the first two to five frames to start the algorithm. Since generally the cameras will see completely different views of the scene, the pose of such frames may be obtained using the essential matrix for each camera respectively, but this method will be ill-conditioned if the initial translations are small. On the other hand, we get the initial pose from the computer program which controls the robot motion. The first frame is usually the reference with zeros in translations and rotation angles, and the next one or four frames respectively (is/ are) accurate enough to start the algorithm since the first few frames do not suffer from drift accumulation yet. The uncertainty of the initialization is tested experimentally in section 4.

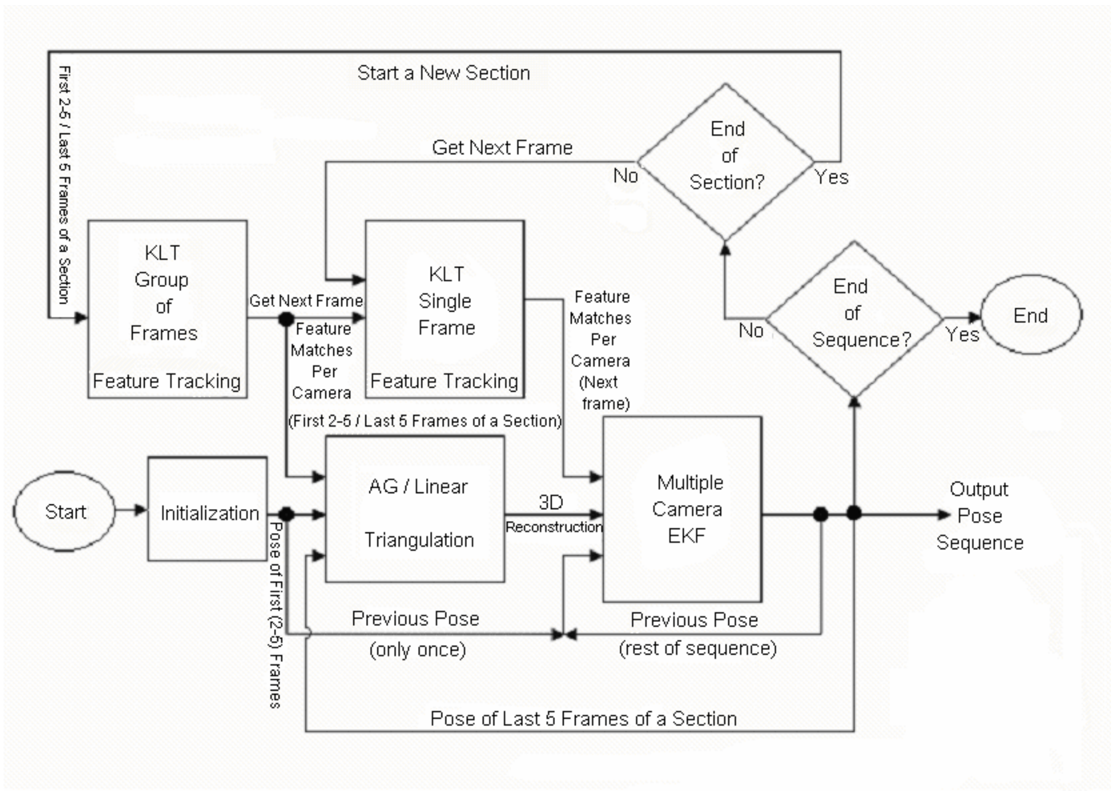
Start a New Section
First 2-5 / Last 5 Frames of a Section
Get Next Frame
No
End of Section?
Yes
KLT Group of Frames Feature Tracking
Get Next Frame
Feature Matches Per Camera
KLT Single Frame Feature Tracking
Feature Matches Per Camera (Next frame)
End of Sequence?
No
Yes
End
(First 2-5 / Last 5 Frames of a Section)
Start
Initialization
AG / Linear Triangulation
3D Reconstruction
Multiple Camera EKF
Output Pose Sequence
Pose of First (2-5) Frames
Previous Pose (only once)
Previous Pose (rest of sequence)
Pose of Last 5 Frames of a Section

Fig.3. Algorithm Flowchart.

3.2.2 Feature Tracking, KLT (Group of Frames)

At this stage we use the KLT algorithm [42] in feature tracking in the first 2-5 frames of the sequence for each camera respectively. However, it is easy to combine any other feature tracker with our algorithm.

3.3.3 AG / Linear Triangulation

The linear and the AG triangulation methods are described in section 2 and Appendix A respectively. This stage uses the outputs of the previous stages namely the pose and the feature correspondences per camera, e.g. for a group of 5 frames, to calculate the 3D reconstruction of such features.

3.3.4 Feature Tracking, KLT (Single Frame)

We move to the next frame for each camera tracking the features whose 3D reconstruction is computed at the previous stage.

3.3.5 Multiple Camera EKF

This is the core stage of our algorithm. The output of this stage is the pose of the current frames of each camera, and the inputs to this stage are the tracked features in them, their 3D reconstruction, and the previous pose which is obtained only once from the initialization stage, then the current output of this stage is fed as the input previous pose for the next time step.

3.3.6 Checking Step

This stage provides the control logic of our algorithm. It first checks if there are upcoming frames in the sequence, if not it stops the algorithm. Otherwise, it checks if a new section should be started, if not the algorithm merely gets the next frame and moves to stage 4, KLT (Single Frame), and continues the processing. If a new section should be started, the last five frames of the current section are fed to stage 2, KLT (Group of Frames) and the processing is continued passing through stage 3, Linear/AG Triangulation which has in this case, as an input, the pose of the last 5 frames of the current section instead of the initial pose of the first (2-5) frames.

The length of a section, as we mentioned above is a trade-off between speed and accuracy; it is known that KLT suffers from drift when tracking long sequences [38], so releasing the features and restarting a new tracking process for the next section will lead to choosing fresh good features to track. On the other hand, starting a new section will lead to the overhead of triangulation. Unlike [38], we do not restart the reconstruction of the new section orthographically, but in contrast, we use the calculated poses in our triangulation.

Although, in some situations when the motion is limited to small translations and angles of rotations, it can be possible to consider all the sequence as one section with the triangulation stage carried only once at the beginning of the sequence, we stick here to sections of 10 frames length to allow the range of motion to increase and to lower the drift of KLT algorithm, and hence the presence of outliers.

3.4 Details of The Multi-Camera EKF Implementation

The EKF inputs are the tracked features in the current frame, their corresponding 3D reconstruction, the previous pose and its covariance. Based on the inputs, the system model, and the statistical description of the system and measurement noise, the EKF can predict the current pose and its covariance which are fed to the EKF in the next frame as the previous ones. Therefore, the EKF is a recursive technique which solves the pose problem frame by frame. In fact, recursive techniques are so popular since they relax the storage requirements and enable real time performance.

3.4.1 State space vector:

$$s = [t_x \quad \dot{t}_x \quad t_y \quad \dot{t}_y \quad t_z \quad \dot{t}_z \quad \alpha \quad \dot{\alpha} \quad \beta \quad \dot{\beta} \quad \gamma \quad \dot{\gamma}]^T \qquad (4)$$

Where t_x, t_y, and t_z are the translations in the corresponding axes directions, $\dot{t}_x, \dot{t}_y$, and $\dot{t}_z$ are their derivatives, α, β, and γ are the roll, pitch, and yaw angles, and $\dot{\alpha}$, $\dot{\beta}$, and $\dot{\gamma}$ are their derivatives.

3.4.2 Plant Equation:

$$s_k = As_{k-1} + n_k \qquad (5)$$

Where the 12×12 matrix A is given by (using MATLAB notation):

$$A = diag\left\{\begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}, \ldots, \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}\right\} \qquad (6)$$

τ is the sampling period, $n_k = N(0, Q)$ is the plant noise, and k is the time step.

3.4.3 Measurement Equation:

$$I_k = h(s_k) + \eta_k \qquad (7)$$

Where I_k is the measurement vector (image points) of *m* different features taken by *n* cameras, for example we can denote (u_i^j, v_i^j) as the image coordinates of the ith feature in the image taken by the jth camera, accordingly:

$$I_k = [u_1^1 \;\; v_1^1 \;\; \dots \;\; u_m^1 \;\; v_m^1 \;\; \dots \;\; u_1^n \;\; v_1^n \;\; \dots \;\; u_m^n \;\; v_m^n]^{\mathrm{T}} \tag{8}$$

The function $h(s_k)$ is the state measurement relation:

$$h(s_k) = \left\{ f_1 \left[\frac{X_1^1}{Z_1^1} \;\; \frac{Y_1^1}{Z_1^1} \;\; \cdots \;\; \frac{X_m^1}{Z_m^1} \;\; \frac{Y_m^1}{Z_m^1} \right], \dots, f_n \left[\frac{X_1^n}{Z_1^n} \;\; \frac{Y_1^n}{Z_1^n} \;\; \cdots \;\; \frac{X_m^n}{Z_m^n} \;\; \frac{Y_m^n}{Z_m^n} \right] \right\}^T \tag{9}$$

X_i^j, Y_i^j, and Z_i^j are the jth camera coordinates of ith feature,

$f_{1, \dots\dots\dots,}\ f_n$ are focal lengths of the n cameras,

And $\eta_k = N(0,\ \lambda)$ is the measurement noise.

3.4.4 Initialization and Filter Tuning:

Initialize $\hat{s}_k(0)$, and $C_k(0)$ (initial estimation of state vector, and initial state covariance matrix respectively). Initial pose is obtained as mentioned in subsection 3.3.1 and the initial velocities, the derivatives, are obtained from the initial pose, say at frame 5 of the whole sequence, and the pose at the preceding frame, frame 4 as follows:

$$\dot{t}_x(0) = \frac{t_x(5) - t_x(4)}{\tau} \tag{10}$$

$\dot{t}_y(0), \dot{t}_z(0), \dot{\alpha}(0), \dot{\beta}(0)$, and $\dot{\gamma}(0)$ are obtained in the same way. Additionally, as we know the initial conditions accurately enough, we can set the elements of the initial state covariance matrix to zeros. Another initialization technique is mentioned in [5]. For the filter tuning, we use the identity matrix for both Q, and λ. Other tuning techniques can be found in [34], [8] and [5].

3.4.5 Prediction:

$$\hat{s}_{\mathrm{k,k\text{-}1}} = \mathrm{A}\hat{s}_{k-1,k-1} \tag{11}$$

$$C_{\mathrm{k,k\text{-}1}} = \mathrm{A}C_{k-1,k-1}\mathrm{A}^T + Q_{k-1} \tag{12}$$

3.4.6 Update:

Use $\hat{s}_{k,k-1}$ obtained from equation (11) in calculating $R_{\mathrm{j}}^{\mathrm{T}}$, and d_j and then substitute in equation (3) for each camera respectively to get the camera coordinates matrix, P_{ijk}.

Use P_{ijk} to get $h(s_k)$ and $J_k = \partial h(s)/\partial s \mid s = \hat{s}_{k,k-1}$. The Jacobian, $\frac{\partial h}{\partial s}$, is a $(2mn \times 12)$ matrix where n is the number of cameras, and m is the number of tracked features. Referring to (9) the state measurement relation h for the k^{th} camera (not to be confused with the k^{th} time step of the filter), the j^{th} frame, and the i^{th} feature is given by:

$$h_{i,j,k} = f_k \left[\frac{X_i^k}{Z_i^k} \ , \ \frac{Y_i^k}{Z_i^k} \right]_j \tag{13}$$

Let $h_U = f_k \frac{X_i^k}{Z_i^k}$, and $h_V = f_k \frac{Y_i^k}{Z_i^k}$ (14)

Then

$$\left[\frac{\partial h}{\partial s}\right]_{i,j,k} = \begin{bmatrix} \frac{\partial h_U}{\partial t_x} & \frac{\partial h_U}{\partial \dot{t}_x} & \frac{\partial h_U}{\partial t_y} & \frac{\partial h_U}{\partial \dot{t}_y} & \frac{\partial h_U}{\partial t_z} & \frac{\partial h_U}{\partial \dot{t}_z} & \frac{\partial h_U}{\partial \alpha} & \frac{\partial h_U}{\partial \dot{\alpha}} & \frac{\partial h_U}{\partial \beta} & \frac{\partial h_U}{\partial \dot{\beta}} & \frac{\partial h_U}{\partial \gamma} & \frac{\partial h_U}{\partial \dot{\gamma}} \\ \frac{\partial h_V}{\partial t_x} & \frac{\partial h_V}{\partial \dot{t}_x} & \frac{\partial h_V}{\partial t_y} & \frac{\partial h_V}{\partial \dot{t}_y} & \frac{\partial h_V}{\partial t_z} & \frac{\partial h_V}{\partial \dot{t}_z} & \frac{\partial h_V}{\partial \alpha} & \frac{\partial h_V}{\partial \dot{\alpha}} & \frac{\partial h_V}{\partial \beta} & \frac{\partial h_V}{\partial \dot{\beta}} & \frac{\partial h_V}{\partial \gamma} & \frac{\partial h_V}{\partial \dot{\gamma}} \end{bmatrix}_j \tag{15}$$

Referring to (3): $h_U = f_k \frac{P_{ijk}(1)}{P_{ijk}(3)}$, and $h_V = f_k \frac{P_{ijk}(2)}{P_{ijk}(3)}$ (16)

Where $P_{ijk}(1)$ is the X-component of P_{ijk}, $P_{ijk}(2)$ is the Y-component of P_{ijk}, and $P_{ijk}(3)$ is the Z-component of P_{ijk}. Let $Rot = R_{\mathrm{k}}^{\mathrm{T}} \ R_{\mathrm{j}}^{\mathrm{T}}$ (17)

Then

$$P_{ijk}(1) = Rot(1,1)[M_i(1) - D_k(1) - t_x] + Rot(1,2)[M_i(2) - D_k(2) - t_y] + Rot(1,3)[M_i(3) - D_k(3) - t_z] \tag{18}$$

Where $Rot(l,q)$ is the element of Rot matrix at the l^{th} row and the q^{th} column, and $d_j = [t_x \quad t_y \quad t_z]^T$ is the translation vector of equation (3). $P_{ijk}(2)$, and $P_{ijk}(3)$ can be written easily in a similar fashion.

Noting that R_k and D_k are constants, $\frac{\partial h_U}{\partial t_x} = f_k \frac{Rot(3,1)P_{ijk}(1) - Rot(1,1)P_{ijk}(3)}{\left(P_{ijk}(3)\right)^2}$ (19)

$$\frac{\partial h_U}{\partial \alpha} = [M_i(1) - D_k(1) - t_x]\frac{\partial Rot(1,1)}{\partial \alpha} + [M_i(2) - D_k(2) - t_y]\frac{\partial Rot(1,2)}{\partial \alpha} + [M_i(3) - D_k(3) - t_z]\frac{\partial Rot(1,3)}{\partial \alpha} \tag{20}$$

Similarly, we can obtain the other elements of the Jacobian, however the details are omitted for the space limits. Moreover, any partial derivatives with respect to velocities such as: $\frac{\partial h_U}{\partial \dot{t}_x}$ $\frac{\partial h_U}{\partial \dot{t}_y}$ $\frac{\partial h_V}{\partial \dot{t}_z}$ $\frac{\partial h_V}{\partial \dot{\alpha}}$, and $\frac{\partial h_U}{\partial \dot{\beta}}$ are zeros since the velocities are not present in equation (3).

Then we go on to calculate the Kalman gain, *G*, and the updated state and updated state covariance as follows:

$$G = C_{k,k-1} J_k^T (J_k C_{k,k-1} J_k^T + \lambda)^{-1} \tag{21}$$

$$\hat{s}_{k,k} = \hat{s}_{k,k-1} + G(I - h_k(\hat{s}_{k,k-1})) \tag{22}$$

$$C_{k,k} = C_{k,k-1} - G J_k C_{k,k-1} \tag{23}$$

3.4.7 Image Feature Selection

We arrange features in each frame according to their distances from the center of the image in an ascending order, and choose a number of them, say 50. This choice realizes several benefits: it is a sort of normalization that satisfies the zero average assumption which is essential for avoiding biased estimates in the EKF [11], the reduction of radial distortion is guaranteed. Furthermore, using such a flexibly changeable set of features renders our algorithm not affected at all by the occlusion of any feature points among frames. Moreover, it may provide a basis for studying the efficiency of our solution in the presence of bas-relief ambiguity as shown in Fig. 4 below.

4. Experiments and Results

4.1 Simulations

The robot carrying the multiple-camera setup was moved as if on the floor with random translations T_x, and T_z in the direction of X and Z axes respectively, and with random rotation angle β around the Y axis. The coordinate system origin coincided with the center of the first camera at the motion start with the Z axis perpendicular at the image frame. The translations were taken randomly with a uniform distribution from ±0.005 to ±0.015 meters, and the rotation angles were taken randomly with a uniform distribution from ±0.005 to ±0.02 radians. A random noise was added to each feature point with a normal distribution of zero mean and 1 pixel standard deviation.

The motion took place inside a spherical surface whose radius is one meter and its center coincides with the origin of the coordinate system. The spherical surface is open from either side in the direction of Y axis extending from -0.22 to 0.22 meters. The feature points were distributed randomly on the spherical surface. The total number of feature points, in the scene, was chosen as 10,000, 15,000 and 20,000 respectively and experiments were carried out to study the accuracy and the speed of the algorithm in each case. Additionally, the different choices of such number is related to the bas-relief ambiguity, since we feed the EKF with a certain number of points, 50, 75, or 100, in each frame which is nearest to the image center, and the larger the total numbers of feature points in the scene, the narrower the spread of the chosen point features around the center of each image. This is equivalent to having a small depth of view such as the case when using a lens of a large focal length to image a distant object with more details and small depth variation.

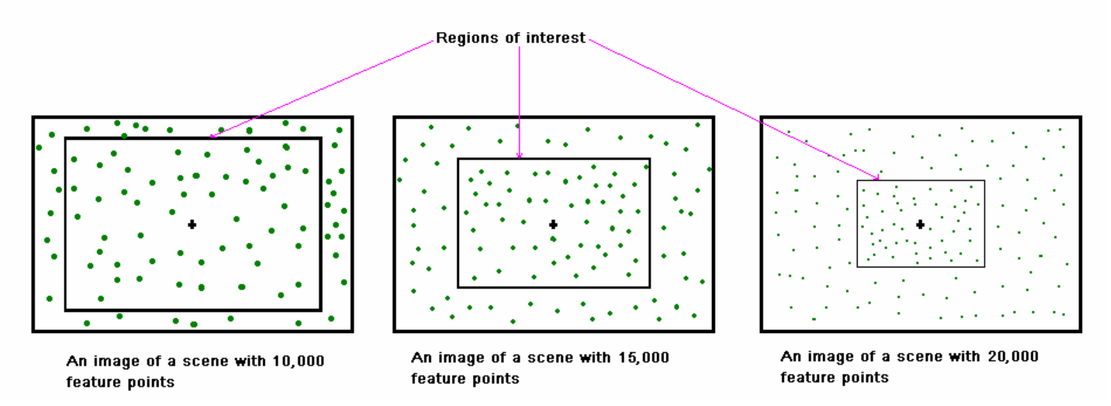
Regions of interest
An image of a scene with 10,000 feature points
An image of a scene with 15,000 feature points
An image of a scene with 20,000 feature points

Fig.4. Approximate drawing showing the relation between the total number of feature points in the scene and the field of view.

We compared different two-camera and single-camera methods. For these sets of experiments we fixed the roll, pitch, and yaw rotation angles of the second camera with respect to the first to [0, π, 0] respectively, and the translations of the second camera center from the first to [0, 0, -0.03] meters in the directions of X, Y, and Z axes respectively at the motion start. Moreover, we carried out two sets of experiments to study respectively the effect of the pose initialization error, and the measurement pixel error of the images on the results.

4.2 Simulation Results

We carried out three sets of experiments with the total number of feature points, in the scene, chosen as 10,000, 15,000 and 20,000 respectively. Each set was run 100 times and for each run the different approaches of the solution were experimented to guarantee a fair judgment. The tests are: using two cameras with linear triangulation, using two cameras with AG triangulation, and then using one camera with linear triangulation having 50 points/frame, 75 points/frame, and 100 points/frame.

The absolute error of the translational and rotational parts of the motion are calculated as well as the time of the processing whether in the EKF part or in the triangulation part of the algorithm. The following three tables give the average of 100 runs, in each case, the absolute errors and the time periods are given per frame, while the convergence rate of each approach is given per the total number of runs. The term: "one frame" is obvious when using a single camera however it means the set of concurrent frames, taken at the same moment, for multiple cameras. All the experiments were run using Matlab-7.0.4 on a machine with a 2.8 MHz Pentium processor, and 512 MB RAM.

Method	T_x Abs. Error meter	T_z Abs. Error meter	β Abs. Error radian	EKF Time second	Triangulation Time second	Convergence %
Linear 2 Cam.	0.000419	0.00119	0.000885	0.01206	0.012205	100
AG 2 Cam.	0.002271	0.004474	0.001917	0.01226	0.074619	92
Linear 1 Cam. 50 points/frame	0.051274	0.04954	0.049933	0.002742	0.006478	100
Linear 1 Cam. 75 points/frame	0.053393	0.059658	0.051274	0.006196	0.006497	100
Linear 1 Cam. 100 points/frame	0.05112	0.070661	0.047674	0.011512	0.006817	100

Table.1. Number of feature points 10,000 /scene

Method	T_x Abs. Error meter	T_z Abs. Error meter	β Abs. Error radian	EKF Time second	Triangulation Time second	Convergence %
Linear 2 Cam.	0.000398	0.001185	0.00095	0.012375	0.018888	100
AG 2 Cam.	0.005386	0.013015	0.004471	1.23E-02	1.14E-01	92
Linear 1 Cam. 50 points/frame	0.045823	0.039463	0.044277	0.002711	0.010253	100
Linear 1 Cam. 75 points/frame	0.044598	0.046363	0.042489	0.006258	0.010365	99
Linear 1 Cam. 100 points/frame	0.046999	0.052829	0.044777	0.011688	0.010199	100

Table.2. Number of feature points 15,000 /scene

Method	T_x Abs. Error meter	T_z Abs. Error meter	β Abs. Error radian	EKF Time second	Triangulation Time second	Convergence %
Linear 2 Cam.	0.000384	0.001267	0.000936	0.012238	0.025238	100
AG 2 Cam.	0.004475	0.009242	0.003064	1.22E-02	1.53E-01	94
Linear 1 Cam. 50 points/frame	0.040851	0.032019	0.040007	0.002737	0.013089	100
Linear 1 Cam. 75 points/frame	0.042712	0.039809	0.041685	0.006286	0.013077	100
Linear 1 Cam. 100 points/frame	0.042179	0.045209	0.040617	0.011638	0.013061	100

Table.3. Number of feature points 20,000 /scene

Note: The pose, with reference to the initial frame, has a sum of absolute translations equal to 1 meter on average, and a sum of absolute rotations equal to 1.25 radians (≈71.6 degrees) on average.

Three random samples of the program runs are shown in Fig.5 below.

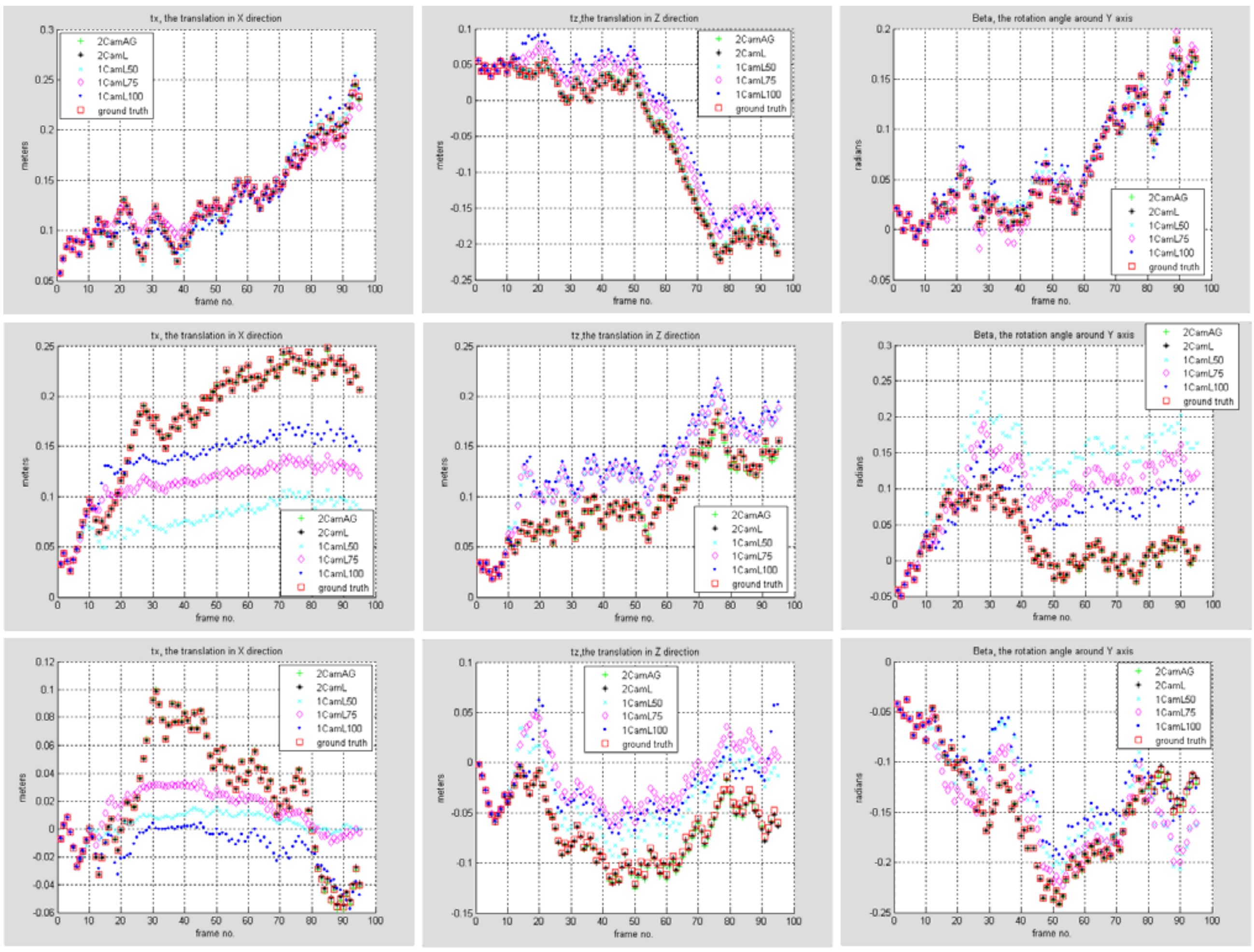

Fig.5. Three samples of the simulation outputs: each row shows t_x, t_z, and β of the same sample obtained by the two-camera, and single-camera methods compared to the ground truth.

Since in real situations, there must be uncertainties in the initialization step due to noise in the pose of the initial frames, we carried out a fourth set of experiments to study the effect of adding Gaussian noise with zero mean and standard deviations ranging from 5% to 50% of each random pose element (translations and rotation angles). These simulations were run for a scene with 10,000 feature points one hundred times for each noise level with a total number of one thousand runs. The results are shown in Fig.6 below. The shown absolute errors are the average per frame, and the error standard deviations are the average per sequence (one run of the program).

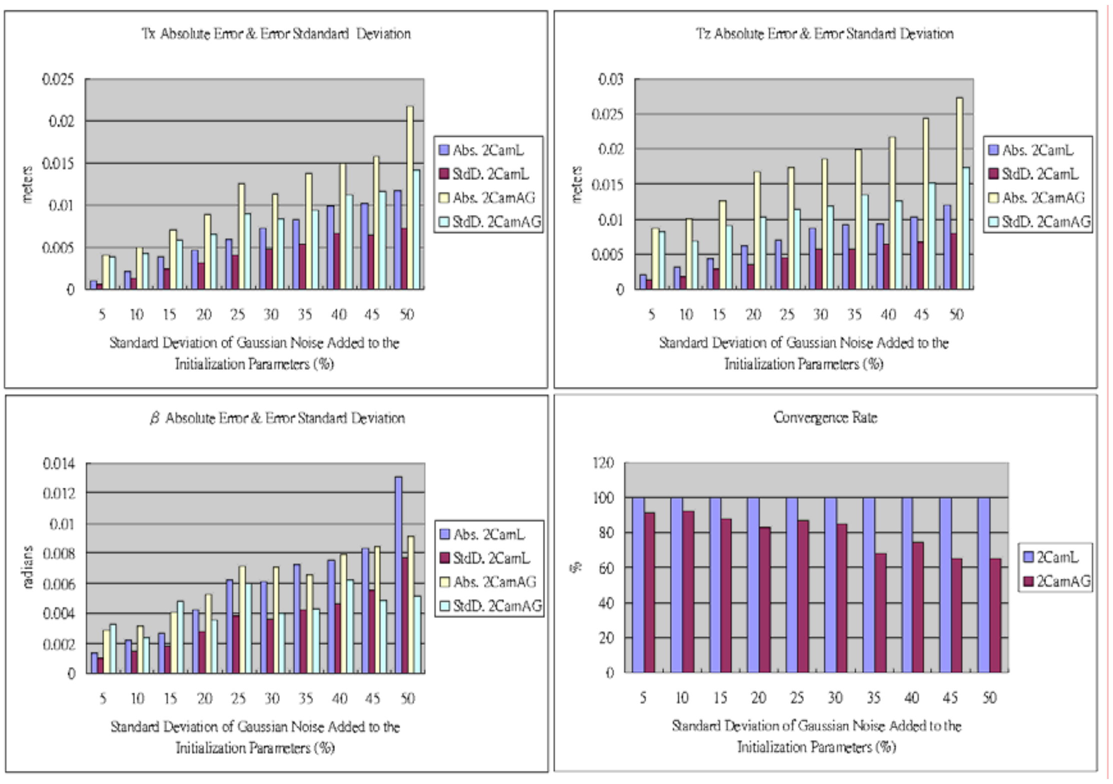

Fig.6. Effect of uncertainties of pose initialization

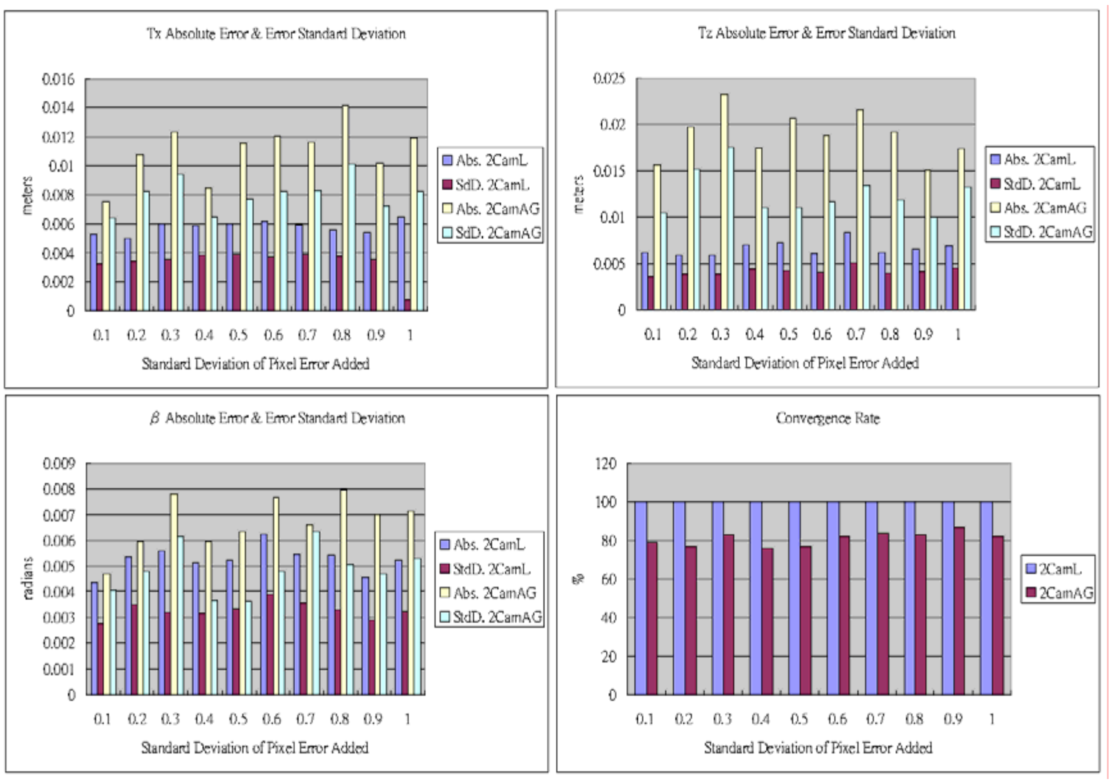

Fig.7. Effect of measurement noise

Moreover, we carried out a fifth set of experiments to study the effect of adding Gaussian noise with zero mean and standard deviations ranging from 0.1 to 1 pixel measurement error to the images (fixing the standard deviation of the initial pose noise to 25%). These simulations were run for a scene with 10,000 feature points one hundred times for each noise level with a total number of one thousand runs. The results are shown in Fig.7 above. The shown absolute errors are the average per frame, and the error standard deviations are the average per sequence (one run of the program).

4.3 Real Experiments

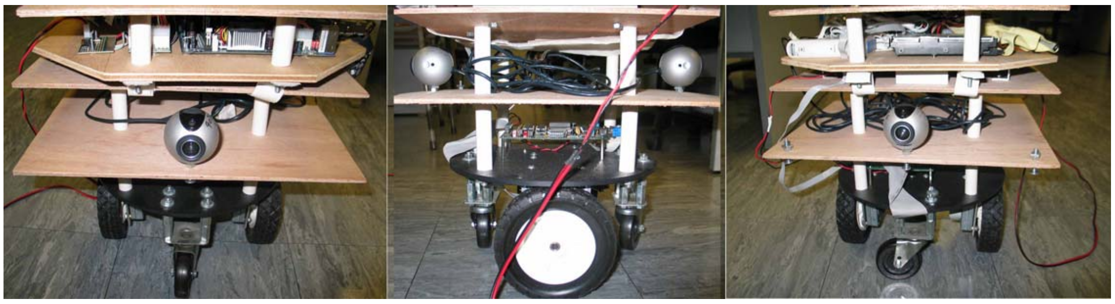

Fig.8. The robot and the cameras: frontal, side, and back views

The robot used is shown in Fig.8. It carries two ordinary web-cameras with 320×240 resolution put back-to-back about 40 centimeters apart. Additionally, the path taken by the robot is defined via a wireless network connected to a computer on-board. Before carrying out the experiments, each camera is calibrated to obtain the internal parameters using [43]. To calibrate the inter-rotation and the translation between the two cameras, we use two upright and parallel check-boards facing each other with a known distance in-between. The extrinsic parameters of each camera with respect to the corresponding check-board are obtained using [43]. Since the relation between the coordinate systems attached to the two check-boards are known, we can easily get the inter-rotation and the translation between the two cameras. We cannot use the stereo-calibration since the two cameras are back-to-back. However, as an alternative, the 1D-calibration mentioned in [21] might be used.

A sequence of 100 frames was taken simultaneously by each camera; some frames are shown in Fig.9. The pose is obtained using the different methods described above and the results are shown in Fig.10, and the ground truth is provided by the computer controlling the robot.

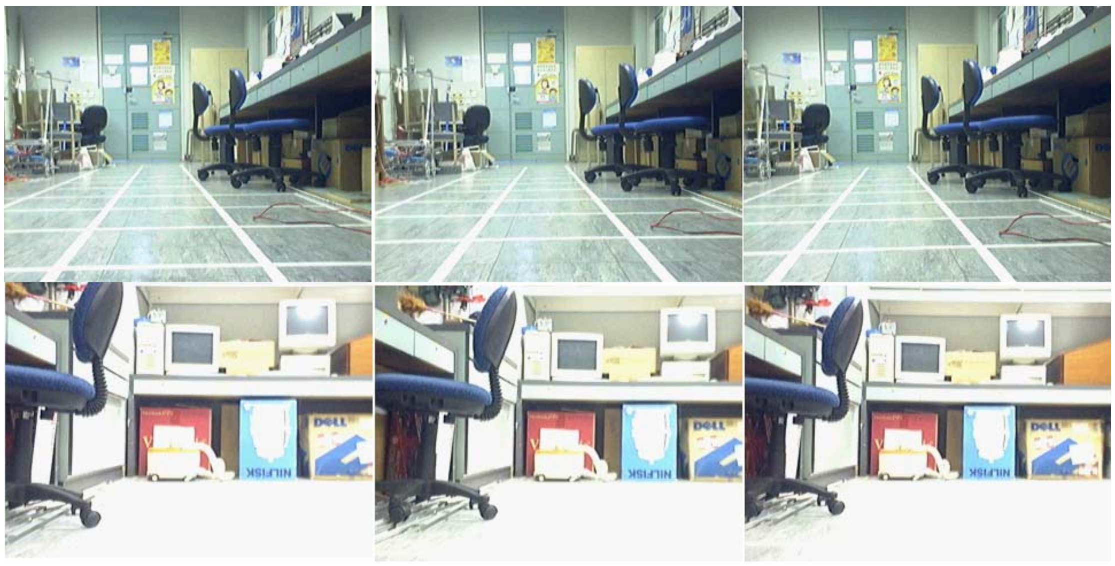

Fig.9.The lab-sequence: each row shows frames 1, 51, and 100 for each camera

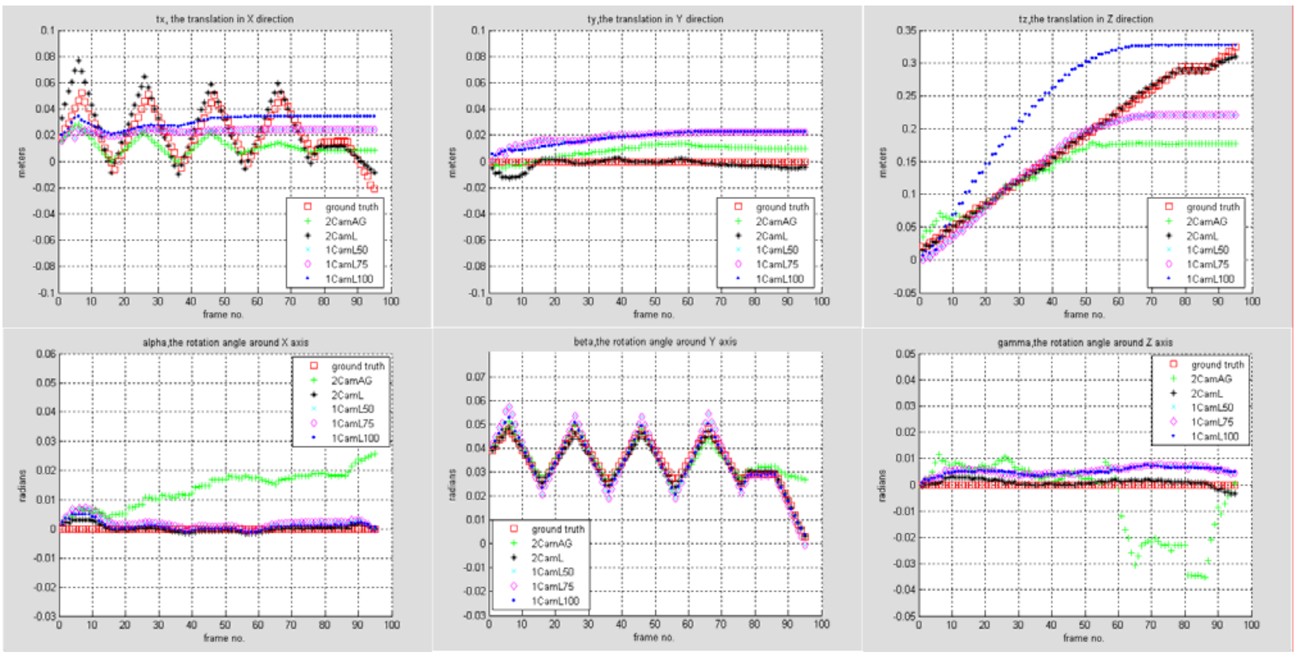

Fig.10. The real experiment results for single and two-camera methods: the upper row shows translations while the lower row shows rotation angles.

5. Discussions

Our EKF algorithm has been used to the following approaches: two cameras with linear triangulation, two cameras with AG triangulation, and single-camera with linear triangulation having 50 points/frame, 75 points/frame, and 100 points/frame respectively. From the absolute error columns in the three tables, it is clear that two-camera methods are far more accurate than single-camera methods and this well justifies the use of multiple cameras rather than one.

5.1 Linear versus AG Triangulation

Both the two triangulation methods are capable of recovering the scale factor as explained in section 2, and appendix A. However, the linear triangulation is better than the AG triangulation in terms of accuracy, speed, and convergence rate. This is expected since the linear triangulation method solves a set of equations simultaneously with SVD to find the best solution in the least squares manner while the AG triangulation finds the midpoint between each pair of skew lines, then takes the average of midpoints which also may suffer from round-off errors. Additionally, the repetition of the AG triangulation algorithm with each pair of the skew lines, ten pairs in case of the five lines used here, is the cause of the slower speed than the linear implementation.

5.2 Single-camera methods

Fixing the total number of feature points in the scene, the three single-camera methods are close to each other in terms of accuracy, and there is no fixed direction of improvement with increasing or decreasing the number of points used in each frame. This indicates that we need more research to find the optimal number of points to use. It is also obvious that, as the total number of points in the scene increases, the accuracy of the single-camera methods slightly increases. On the other hand, the required time increases as we move down from a table to another with a larger number of total feature points in the scene, and within the same table from a row to another with a larger number of feature points per frame.

5.3 EKF Time versus Triangulation Time

There are two entries for time in the three tables, one for the time spent in the multiple-camera EKF step, and the other for the time spent in the triangulation step. Both times are given per frame. For example, looking at the first row of the first table, we find out the average time required per frame in the two-camera method with linear triangulation is the summation of these two times, 0.024265 seconds. Therefore this algorithm can process about 41 frames per second with a very high degree of accuracy. Furthermore, even more frames can be processed per second by relaxing the triangulation overhead, for example, it can be carried out once in each set of 20 frames not 10 frames as in the current implementation; this tradeoff between speed and accuracy was explained in section 3. Practically, this high speed will be limited by the maximum frame rate the hardware used can handle, the image size, the feature tracker used, and the number of features to be tracked. However, since tracking a few number of features, 50 or lower, is enough for accurate pose estimation, the real time performance is guaranteed. Additionally, further speed can be gained by using multiprocessing; each camera can have its processor to track features and triangulate whenever needed. With respect to the EKF time, it is approximately constant for the same method in the three tables, since the main time consuming of this approach is a matrix inversion, see equation (21), which depends on the number of points used for each frame. Therefore, the time required by the EKF algorithm increases in case of the single-camera methods from the 50 points method to the 100 points which is slightly faster than the EKF algorithm in the case of the two-camera methods which use also 100 points. The slight increase in EKF time in case of the two-camera methods is due to the formulation of the geometric relations between the two cameras. Again, the EKF times for the two-camera methods are quite close in all tables to further prove our conclusion.

With respect to the triangulation time of the simulations, we triangulate all available features at the beginning of each section to deal with motion randomness. That is why the linear triangulation time is nearly constant for the three single-camera methods in each table regardless of the number of points used. However as we move down to the next table where the number of feature points increases, the linear triangulation time increases. The time required for the two-camera method with linear triangulation in each table is nearly the double of any single-camera method, since the triangulation algorithm is carried out in this case twice, one for each camera frame. When moving from a table to the next, the AG triangulation nearly increases by the same ratio as the linear triangulation however for each table, it is longer than the linear triangulation method as explained in subsection 5.1.

5.4 The Two-camera methods and the Number of Points in the Scene

Now, we compare each two-camera method among the three tables. The two-camera method with linear triangulation has nearly the same accuracy in all tables which is a good indication that it can be used in resolving bas-relief ambiguity. Its convergence rate is 100% in all cases, however, the required time for the triangulation increases as the number of feature points increases which is the case in simulation only since we triangulate all available features at the triangulation step. In real circumstances, the triangulation is carried out for a fixed number of features.

With respect to the two-camera method with AG triangulation, it is obvious from the three tables that the convergence rate is not affected by narrowing the field of view. Generally, the accuracy of Table.1 results is the highest and that of Table.2 is the lowest indicating again that there is no fixed direction of improvement and more research should be done to find the optimal settings.

5.5 The Effect of Uncertainties in Initial Pose

As shown in Fig.6, fixing the method used, the absolute errors and the error standard variations of pose components generally increase with the increase of the percentage of the standard deviation of the Gaussian noise added to the initial pose. This is expected since the initial reconstruction is affected and the error propagates to subsequent poses. Therefore, we need to initialize the algorithm as precise as possible to guarantee accurate results.

5.6 The Effect of Measurement Noise

As shown in Fig.7, while the standard deviation of the average-mean Gaussian noise added to the images varies from 0.1 to 1 pixel and the method used is fixed, the pose absolute errors and standard deviation errors vary slightly and can be considered almost constant. This agrees completely with the theoretical foundation of the Kalman Filter which can estimate the state space vector accurately under the assumption of average-mean Gaussian measurement noise [19]. We think that the filter would perform well even if the standard deviation of the measurement error increased much more, albeit it might need re-tuning.

5.7 Real Experiments

As shown in Fig. 10, although the ground truth is not perfect due to possible drift and slipping of the robot wheels, the two-camera method with linear triangulation follows it fairly well. The slight variations of t_y, α, and γ around the ground truth are caused by small perturbations in the position of the robot center-of-gravity due to the presence of the small wheels shown in Fig. 8 (they are used only for supporting the robot while the motion is transferred from the motor-assembly to the larger wheels via a gear-box). Although, the two-camera method with AG triangulation is closer to the ground truth than single-camera methods in cases of t_x and t_y, it suffers from drift especially in cases of $t_z, \alpha,$ and γ. We believe that using accurate cameras with resolutions higher than 320×240 provided by the web-cameras used will allow more precise calibration and provide more accurate results.

6. Conclusions and Future Work

In this work, we introduce a novel approach, using multiple-camera EKF, to find the motion of the set of cameras, or rather the motion of the robot carrying them. The novelty of this approach is the use of the multiple cameras, two in this work, have a large rotation angle in-between, and do not need to have any common features across the camera at the same time. Additionally, the features are tracked from each frame to the subsequent of the same camera which realizes a small baseline and allows accurate tracking which much reduces the outliers. Using a flexibly varying set of points for each frame, the nearest to the image center, to be fed to the EKF algorithm stabilizes the filter and again reduces the effect of outliers which tend to move largely away from their false correspondences, and if they are close to the image center, the negative effects will be minimal.

The AG triangulation method has been compared to the linear triangulation method. Furthermore, both the two-camera methods have been compared to three single-camera methods with different point numbers of features per frame, and different point numbers in the scene which has been related to the small depth of view of the bas-relief ambiguity. Additionally, we have studied the effects of the uncertainties in the initial pose, and the measurement noise on the accuracy and the stability of the two-camera methods. Moreover, we have compared all methods in a real experiment setting.

The accuracy, the speed, and the stability of the two-camera method with linear triangulation render this method quite suitable for precise real time applications.

For future work, we would carry out more research to study the case of more than two cameras, the translations and rotations among them to find the optimal settings. Additionally, we would study the bas-relief ambiguity in more details under this setting.

Self-initialization of the initial pose would be studied perhaps by adding a camera or more to the assembly to serve two purposes: providing accurate self-initialization, and recovering pose optimally. Furthermore, we would combine the linear and the AG triangulation methods to make use of the accuracy of the former and the potential outlier-removing capability of the later in wide baseline settings. Moreover, we would try other feature trackers which may be more efficient than the KLT. Finally, we would extend the algorithm to obtain an accurate 3D reconstruction of the scene on the fly.

Appendices

A. Analytic Geometry (AG) Triangulation

If we have two known camera centers C, C' with respect to a global reference (coordinate system), and two image points x, x' (in two camera frames) for one 3D point *X*, then we can refer both of x and x' to the global coordinate system then get the point of intersection of the rays Cx and $C'x'$. Certainly, these two rays will be skew due to noise however we can get the midpoint between them since the cameras are calibrated (Euclidean setting). We can extend the AG triangulation to any number of images; we choose 5 throughout this work. In this case, we get 10 midpoints (ray 1 with rays 2, 3, 4, and 5- ray 2 with rays 3, 4, and 5 and etc.), and the resultant midpoint is the average of them all.

The algorithm for calculating the midpoint of skew rays is a major modification of the algorithm in [41] which can be used only for two line segments. Fig.11. below describes the AG Triangulation. Normally, we apply this method to a set of five cameras however the three cameras shown in the figure explain the method well. All camera centers: C , C' , and C'' have known locations, and all image planes have known rotations with reference to the global coordinate system at *O*. Now, we will calculate the coordinates of the image point *x* in the image plane of the first camera.

With reference to the coordinate system at the first camera center *C*:

$$x_C = \left[u \times s_x, v \times s_y, f\right]^T \qquad \text{(A1)}$$

Where (u, v) are the image coordinates in pixels of x, (s_x, s_y) are respectively the horizontal and vertical distances, in the image plane, between any two adjacent pixels in meters, and f is the focal length of the first camera in meters.

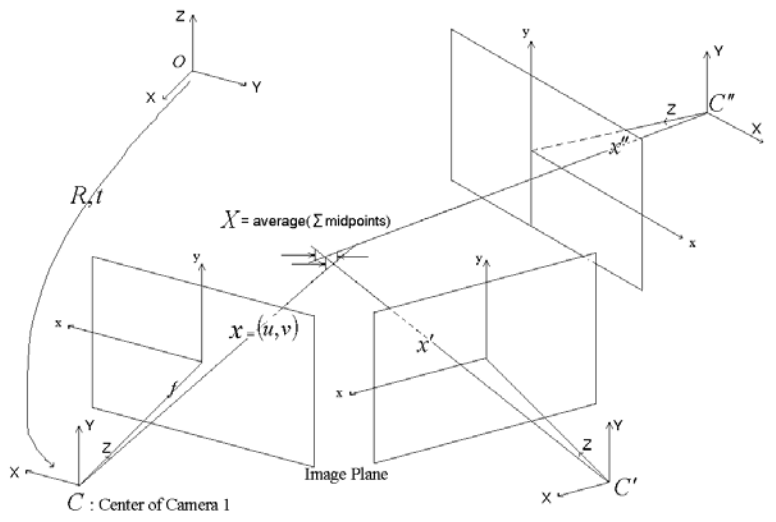

Fig.11. AG Triangulation

With reference to the global coordinate system at *O*:

$$x_O = Rx_C + t \qquad \text{(A2)}$$

Where *R*, and *t* are respectively the rotation and translation of the first camera center, *C*, with reference to the global coordinate system at *O*. Global coordinates of x', and x'' can be obtained in the same way.

What we note about this solution is that it is capable of resolving the scale factor ambiguity since the camera centers are known with reference to a global coordinate system. Additionally, the outliers can be detected and removed easily if we assume for example that if a midpoint is far from the average (by $3\times$standard deviation of the midpoints in the direction of any coordinates axis), it will be considered as an outlier. Furthermore, if the midpoint occurred behind the camera centers, which is detected by a sign reversal, the correspondence will be due an outlier.

B. Using a Fixed Coordinate System in the Scene

As mentioned in subsection 3.2, we will consider here the first camera starts from non-zero rotation angles and hence non-identity rotation matrix. Referring to Fig.1, assume that at the time of calibrating the rotation and translation of the k^{th} camera with respect to the first we find that:

The rotation matrix of the first camera around a reference coordinate system is r_1calib, the rotation matrix of the k^{th} camera around the same coordinate system is r_kcalib, and the translation vector of the k^{th} camera from the first, with reference to the same coordinate system, is D_kcalib.

Then, R_{1k}, the inter-rotation between the k^{th} camera and the first (as if the first has an identity rotation matrix) is given by:

$$r_1calib \times R_{1k} = r_kcalib \qquad \text{(B1)}$$

Multiplying both sides by r_1calib^T which is the transpose and inverse of r_1calib:

$$R_{1k} = r_1calib^T \times r_kcalib \qquad \text{(B2)}$$

Now, if R_1 the initial rotation matrix of the first camera is different from that at the calibration time then, following (B1), the initial rotation of the k^{th} camera is:

$$R_k = R_1 \times R_{1k} \qquad \text{(B3)}$$

Additionally, if $R_{relative}$ is the relative rotation between R_1 and r_1calib such that:

$$R_1 = R_{relative} \times r_1calib \qquad \text{(B4)}$$

Then $R_{relative} = R_1 \times r_1calib^T$ (B5)

And accordingly D_k, the initial translation vector of the k^{th} camera will be:

$$D_k = R_{relative} \times D_kcalib \qquad \text{(B6)}$$

Now that we have computed R_k and D_k in (B3) and (B6) respectively, we can feed both of them into equation (3) and start processing.

References

[1] A. Azarbayejani and A.P. Pentland, "Recursive estimation of motion, structure, and focal length", IEEE Trans. on PAMI, vol. 17, no 6, June 1995.

[2] M. Baeg, H. Hashimoto, F. Harashima, and J. B. Moore, "Gradient Flow Approach for Pose Estimation of Quadratic Surface", Industrial Electronics, Control, and Instrumentation, 1995., Proceedings of the 1995 IEEE IECON 21st International Conference on, pp. 161 - 166 vol.1, 6-10 Nov. 1995.

[3] Baker, P., Fermuller, C., Aloimonos, Y., and Pless, R., "A spherical eye from multiple cameras (makes better models of the world)", Computer Vision and Pattern Recognition, 2001, vol 1, 8-14 Dec. 2001, pp. 576-583.

[4] Y. Bar-Shalom and X.-R. Li, Estimation and Tracking: Principles, Techniques and Software. Norwood, MA: Artech House, 1987.

[5] Y. Bar-Shalom, and X. R. Li, and T. Kirubarajan, Estimation with Applications to Tracking and Navigation Theory Algorithms and Software, John Wiley, 2001.

[6] P.A. Beardsley, A. Zisserman, and D. W. Murray, " Sequential Updating of Projective and Affine Structure from Motion", International Journal of Computer Vision 23(3), 235-259, 1997.

[7] G. A. Borges, and M. Aldon, "Optimal Mobile Robot Pose Estimation Using Geometrical Maps", Robotics and Automation, IEEE Transactions on, Volume 18, Issue 1, Feb. 2002.

[8] T. J. Broida, S. Chanrashekhar, and R. Chellappa, " Recursive 3-D Motion Estimation from a Monocular Image Sequence," IEEE Trans. Aerospace and Electronic Systems, vol. 26, no. 4 July 1990.

[9] M.F.M. Campos, and L.S. Coelho, "Autonomous Dirigible Navigation Using Visual Tracking and Pose Estimation", Proceedings of the 1999 IEEE International Conference on Robotics& Automation, Detroit, Michigan, May 1999.

[10] W. Chang, and C. Chen, "Pose Estimation for Multiple Camera Systems", Proceedings of the 17th (ICPR'04), pp. 262 - 265 Vol.3, 23-26 Aug. 2004.

[11] A. Chiuso, P. Favaro, H. Jain, and S. Soatto, "Structure from Motion Causally Integrated Over Time," IEEE Trans. on PAMI, vol. 24, no 4, April 2002.

[12] M. Grossberg and S. Nayar, "A general imaging model and a method for finding its parameters," Proc. Int. Conf. Computer Vision, 2001.

[13] R. M. Haralick, H. Joo, C. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose Estimation from Corresponding Point Data", IEEE Trans. On Systems, Man, And Cybernetics, vol. 19, No. 6, November/December 1989.

[14] R. Hartley and A. Zisserman, Multiple View Geometry in computer vision, Cambridge University Press, 2003.

[15] K.S. Huang, and M.M. Trivedi, "Robust Real-Time Detection, Tracking, and Pose Estimation of Faces in Video Streams, Pattern Recognition, 2004, ICPR 2004, Proceedings of the 17th International Conference on, pp. 965 - 968 Vol.3, 23-26 Aug. 2004.

[16] M. Isard, and A. Blake, "Contour tracking by stochastic propagation of conditional density", In Proc. European Conf. Computer Vision, 1996, pp. 343-356, Cambridge, UK.

[17] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. NewYork: Academic, 1970.

[18] V. Lippiello, B. Siciliano and L. Villani, "Position and Orientation Estimation Based on Kalman Filtering of Stereo Images", Proceedings of the 2001 IEEE International Conference on Control Applications September 5-7, 2001.

[19] P.S. Maybeck, Stochastic Models, Estimation, and Control, vol. 1., New York, Academic Press, 1979.

[20] D.P. McReynolds and D. Lowe,"Rigidity Checking of 3D Point Correspondences under Perspective Projection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, pp. 1174-1185, 1997.

[21] G. Medioni, and S.B. Kang, Emerging Topics in Computer Vision, NJ, USA, Prentice Hall, 2004.

[22] L. P. Morency, A. Rahimi, and T. Darrell, "Adaptive viewbased appearance model", In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[23] C. Nakajima, and N. Itho, "A Support System for Maintenance Training by Augmented Reality", Image Analysis and Processing, 2003.Proceedings, 12th International Conference on, pp.158 – 163, 17-19 Sept. 2003.

[24] D. Nistér, "An Efficient Solution to the Five-Point Relative Pose Problem", IEEE Trans. On PAMI, vol. 26, No. 6, June 2004.

[25] R. Pless, "Using many cameras as one," Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, 2003.

[26] I. D. Reid and D. W. Murray. Active tracking of foveated feature clusters using affine structure. International Journal of Computer Vision, 18(1):41-60, 1996.

[27] S.D. Roy, S. Chaudhury, and S. Banerjee, "Recognizing Large Isolated 3-D Objects Through Next View Planning Using Inner Camera Invariants", IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics , vol. 35, no. 2, April 2005.

[28] D. C. Schuurman, and D. W. Capson, "Direct Visual Servoing Using Network-synchronized Cameras and Kalman Filter", Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington, DC., May 2002.

[29] A. Shademan, F. Janabi-Sharifi, "Sensitivity analysis of EKF and iterated EKF pose estimation for position-based visual servoing", Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference on, pp. 755 – 760, 28-31 Aug. 2005.

[30] R. Szeliski, and S. B. Kang, "Shape Ambiguities in Structure From Motion", IEEE Trans. On PAMI, vol. 19, No. 5, May 1997.

[31] S. Tariq and F. Dellaert, "A Multi-Camera 6-DOF Pose Tracker", Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium, 2 -5 Nov., 2004, pp. 296-297

[32] C. Tomasi and and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. International Journal of Computer Vision, 9(2):137-154, November 1992.

[33] W. Triggs, P. F. McLauchlan, R. I. Hartley, and A. Fitzgibbon. Bundle adjustment for structure from motion. In Vision Algorithms: Theory and Practice. Springer-Verlag, 2000.

[34] E. Trucco, and A. Verri, Introductory Techniques for 3-D Computer Vision, New Jersey, Prentice Hall, 1998.

[35] A.T. Tsao, Y.P. Hung, C.S. Fuh, Y.S. Chen, "Ego-motion Estimation Using Optical Flow Fields Observed FromMultiple Cameras", in CVPR 1997, pp. 457-462.

[36] J. Weng, N. Ahuja, and T.S. Huang, " Optimal Motion and Structure Estimation," PAMI, vol. 15, No. 9, September 1993.

[37] M. Yang, Q. Yu, H. Wang, and B. Zhang, "Vision based Real-Time Pose Estimation for Intelligent Vehicles", 2004 IEEE Intelligent Vehicles Symposium, Parma, Italy, June 14-17, 2004

[38] Y. K. Yu, K.H. Wong, and M. Chang, "Recursive Three-Dimensional Model Reconstruction Based on Kalman Filtering", IEEE Trans. On Systems, Man, And Cybernetics, vol. 35, No. 3, June 2005.

[39] X. Zhang, N. Navab, "Tracking and Pose Estimation for Computer Assisted Localization in Industrial Environments", Applications of Computer Vision, 2000, Fifth IEEE Workshop on, pp. 214 – 221, 4-6 Dec. 2000.

[40] X. Zhuang, and Y. Huang, "Robust 3D-3D Pose Estimation", Computer Vision, 1993. Proceedings., Fourth International Conference on, pp. 567 – 571, 11-14 May 1993.

[41] http://sun.uni-regensburg.de/idl-5.5/html/idl4/jhuapl.doc/vectors/v_skew.html

[42] http://www.ces.clemson.edu/~stb/klt/

[43] http://www.vision.caltech.edu/bouguetj/calib_doc/