Published in Presence: Teleoperators and Virtual Environments 11(5), pp. 474-492, MIT Press, Oct. 2002

3-D Motion and Structure Estimation Using Inertial Sensors and Computer Vision for Augmented Reality

Authors:

Lin Chai
Xilinx Corp.
2300 55th St.
Boulder, CO 80301
303-544-5521
Lin.Chai@xilinx.com

William A. Hoff (corresponding author)
Engineering Division
Colorado School of Mines
Golden, CO 80401
303-273-3761
whoff@mines.edu

Tyrone Vincent
Engineering Division
Colorado School of Mines
Golden, CO 80401
303-273-3641
tvincent@mines.edu

3-D Motion and Structure Estimation Using Inertial Sensors and Computer Vision for Augmented Reality

Abstract

A new method for registration in augmented reality (AR) was developed that simultaneously tracks the position, orientation, and motion of the user's head, as well as estimating the three-dimensional (3-D) structure of the scene. The method fuses data from head-mounted cameras and head-mounted inertial sensors. Two Extended Kalman Filters (EKF) are used; one of which estimates the motion of the user's head and the other that estimates the 3-D locations of points in the scene. A recursive loop is used between the two EKFs. The algorithm was tested using a combination of synthetic and real data, and in general was found to perform well. A further test showed that a system using two cameras performed much better than a system using a single camera, although improving the accuracy of the inertial sensors can partially compensate for the loss of one camera. The method is suitable for use in completely unstructured and unprepared environments. Unlike previous work in this area, this method requires no *a priori* knowledge about the scene, and can work in environments where the objects of interest are close to the user.

Index terms: Augmented reality, pose estimation, registration, Kalman filter, structure from motion, computer vision, inertial sensors

1 Introduction

This paper describes a new method for registration in augmented reality (AR) that simultaneously tracks the position, orientation, and motion of the user's head, as well as estimating the three-dimensional (3-D) structure of the scene. The method is suitable for use in completely unstructured and unprepared environments. Unlike previous work in this area, this method requires no *a priori* knowledge about the scene, and can work in environments where the objects of interest are close to the user.

The term "augmented reality", as used in this paper, refers to systems that draw visual overlays on top of the user's view of the real world (see (Azuma 1997) for a survey). This is often accomplished using see-through head-mounted displays (HMD's), although it could include hand-held devices. The virtual objects are displayed such that they are registered with corresponding real world objects, to give the sense that the virtual and real objects co-exist in the same space. To preserve this illusion, the virtual objects must be accurately registered to the real world, even as the user moves about the scene. This requires accurate sensors to measure the 6 degree-of-freedom (DOF) position and orientation (pose) of the user's head in real time.

In this paper, we are particularly interested in environments that are unprepared, in the sense that we have not pre-placed sensors, markings, or other objects to aid in registration. Also, we may have no *a priori* knowledge about the environment, in terms of the types of objects that are present or their locations. These are characteristics of many potential AR applications in maintenance, navigation, military, and law enforcement.

As an example, consider a scenario where a maintenance worker, wearing an AR system, enters an unfamiliar room to fix a piece of equipment. The AR system would automatically track the pose of the worker's head in the room, with respect to some arbitrary (but fixed) origin

in the room. Image snapshots could be sent to a remote expert, who could annotate the piece of equipment with text or graphics. These annotations would be fixed in the coordinate system of the room, so that they remained registered with the object of interest. The requirements for this "tele-maintenance" scenario are similar to many other potential applications of remote collaboration.

Tracking the user's head in unstructured environments precludes the use of sensors that require pre-placing instruments, transponders, or landmarks in the scene. These include most mechanical, magnetic, acoustic, and optical sensors (Meyer 1992). One possible alternative is inertial sensors (gyroscopes and accelerometers), but these only provide rate information, not absolute pose. As a result, the derived pose will drift over time. The orientation drift can be corrected by occasional updates from an absolute sensor such as a compass and tilt sensor (Foxlin 1996) (Azuma, Hoff et al. 1999). Also, translation information may be obtained from a Global Positioning System (GPS) sensor (Azuma, Hoff et al. 1999). However, compasses and GPS systems may be unreliable indoors.

A promising alternative is to use head-mounted cameras and computer vision techniques to locate and track naturally occurring features in the scene (Neumann and You 1999). The pose of the camera, as well as the positions of the features in 3-D, can be estimated from the projections of the features in the images.

However, computer vision alone may not be sufficient to meet the demands of accuracy, reliability, and real-time operation needed in AR systems. Computer vision algorithms are computationally intensive and it may be difficult to produce results at a rate needed to keep up with rapid head motion. Also, features are sometimes not detected due to occlusions, lighting changes, or motion blur. Therefore, a hybrid system, incorporating multiple sensors, will

probably be necessary (You, Neumann et al. 1999). For example, inertial sensors would complement head-mounted cameras. Computer vision would provide absolute pose information, while the inertial system would provide pose estimates in between vision updates and when vision data was not available. Such a sensor suite would still be suitable for use in unstructured environments.

The main issues we address in this paper are as follows:

- How can we simultaneously estimate both the 3-D positions of naturally occurring features, as well as the pose of the head, without any *a priori* information? It is relatively straightforward to compute pose from visual observations, if one knows the true 3-D locations of observed features (see, for example, (Hoff, Lyon et al. 1996), (Sharma and Molineros 1997)). It is also relatively straightforward to compute the 3-D locations of observed features if one knows the true motion of the camera (see, for example, (Hoff and Sklair 1990), (Shekhar and Chellappa 1992), (Bhanu, Das et al. 1996)). However, since we don't know one before the other, we must compute both simultaneously.
- How can we model the dynamics of the user's motion, and estimate this from sensor (vision and inertial) data? A motion model is needed to predict the future pose of the head. This can greatly improve dynamic accuracy (Azuma and Bishop 1994), and also improve the efficiency of the computer vision system, by predicting the locations of features in the images.
- How can we fuse vision and inertial sensor data together, when each may be obtained at different update rates?

The main contribution of this paper is the description of a method that addresses the issues described above, and the presentation of results of testing our algorithm implementation

on synthetic and real data. We do not address in this work issues such as detecting and tracking the visual features, sensor calibration, and real-time implementation issues. Most of the material in this paper comes from the M.S. thesis of one of the authors (Chai 2000).

The remainder of this paper is organized as follows. Section 2 provides a background on previous related work, and sets the context of our own work. Section 3 describes the algorithm that was developed. Section 4 describes our experimental augmented reality system that was used to test the algorithm, as well as the software design. Section 5 illustrates the application of the method to various synthetic and real data sets, using off-line (non real-time) processing. Finally, Section 6 provides a discussion.

2 Previous Work

The problem of estimating self motion and self pose has been well studied in the robotics and computer vision research communities, and more recently, in the AR community. Again, we are most interested in approaches that do not modify the environment, and make no *a priori* assumptions about the environment.

2.1 Mobile Robotics

In the mobile robotics field, many approaches have been developed to support navigation in unstructured environments (Borenstein, Everett et al. 1996). Often the robot will build a local map of its environment and estimate its pose with respect to that map. Methods have been developed to fuse sensor data using occupancy grids (Moravec 1988) and Kalman filters (Kriegman, Triendl et al. 1989) (Barshan and Durrant-Whyte 1995). However, many approaches use sensors that are unsuitable for AR applications, due to size, weight, and cost. These include high-end inertial navigation systems (INS) (Bhanu, Das et al. 1996) (Sammarco 1994), lidar

(Weiß, Wetzler et al. 1994) (Forsberg, Larsson et al. 1995), odometry, sonar, millimeter wave radar, *etc*. Other approaches take advantage of the fact that the robot's motion is constrained, for example, to lie on a plane (Crowley 1996) or on a road (Dickmanns and Mysliwetz 1992). In AR, however, the user's motion is unconstrained and not under the control of the AR system.

2.2 Computer Vision

In the computer vision field, much work has been done to estimate motion and structure from image sequences alone (see, for example, (Tomasi and Kanade 1992) (Morita and Kanade 1997) (Debrunner and Ahuja 1998) (Soatto and Perona 1998)). The term "motion" usually refers only to the relative pose of the camera between successive images – it does not usually refer to the dynamics of the camera's motion in terms of explicit velocities or accelerations. The term "structure" refers to the 3-D locations of features, assumed to lie on a rigid body. Here, we focus on methods that track a discrete set of features in the image, as opposed to "optical flow" based methods that compute a dense velocity flow field across the entire image.

Although the problem can be solved using a single camera alone, there is an unknown scale factor in the resulting estimated camera and point positions. Some additional information is necessary to recover the unknown scale. This could be data from a second camera (which can provide absolute point positions from stereo triangulation), data from an INS, or observation of an initial set of landmarks with known 3-D geometry.

A minimum of five point correspondences between two perspective images is necessary to compute motion and structure (Faugeras 1993). However, the resulting estimates are very sensitive to noise in the observed image points (Fang and Huang 1984). Rather than using only two views, a larger number of image frames can be used to improve accuracy. For AR, we are

more interested in recursive techniques, such as the Kalman filter, than batch techniques, due to the requirements of real-time operation.

The Extended Kalman filter (EKF) (Brown and Hwang 1992) has been used by a number of computer vision researchers. Ayache and Faugeras have developed a number of systems that compute camera motion as well as the 3-D positions of point and line features (Ayache and Faugeras 1988) (Ayache and Faugeras 1989) (Faugeras 1993). They use separate state vectors for each feature to be tracked, as well as the motion of the camera. Again, "motion" refers only to the relative pose of the camera between frames. Similar work includes that of Matthies and Shafer (Matthies and Shafer 1987). In these works, there is an explicit representation of the error of each feature point in the form of a 3-D Gaussian distribution

Broida, Chandrashekhar, and Chellappa developed an EKF with a dynamic motion model that includes position, orientation, angular velocity, and translational velocity. A single state vector was used for the motion as well as the 3-D positions of each of the feature points. The state vector has 3N+12 elements, where N is the number of feature points being tracked. Since a single camera is used, there is still an unknown scale factor. The authors report that the algorithm is sensitive to the accuracy of the initial guess of the state vector.

Azarbayejani and Pentland (Azarbayejani and Pentland 1995) extended the work of Broida, *et al*, to include estimating camera focal length as well as motion and structure (although they do not include translational velocity and acceleration in their state vector). However, instead of using three parameters per point (*i.e.*, XYZ), they use only a single parameter for each point – its depth. This reduces the dimensionality of the problem, at the expense of losing the explicit representation of the 3-D uncertainty of each point's location.

2.3 Augmented Reality

In the augmented reality field, much work has been done recently on registration using inertial sensors and head-mounted cameras.

Foxlin (Foxlin 1996) developed a system to track head orientation using a combination of angular rate sensors, inclinometers, and compasses. He used a Kalman filter, but instead of predicting head dynamics, his approach was to estimate the sensor characteristics. The Complementary Kalman Filter (CKF) was used as an error estimator to predict gyroscope biases. The integration of the Euler angles was carried out outside of the Kalman filter as a separate step. In later work (Foxlin, Harrington et al. 1998), he combined this with accelerometers and ultrasonic range sensors to allow translation measurement as well as orientation.

Instead of estimating sensor characteristics with a Kalman filter, Azuma (Azuma and Bishop 1994) took the approach of estimating head dynamics. He developed a system to track 6 DOF head pose using angular rate sensors, accelerometers, and an optical tracker. Three separate linear Kalman filters were used to estimate translation, velocity, and acceleration for each axis of the head position. A single EKF was used to estimate head orientation, angular velocity, and angular acceleration.

The previous works used tracking sensors that required placing active targets (powered emitters and transponders) in the environment; whereas in many applications it is preferable to use passive targets (ambient or naturally occurring signals) (You, Neumann et al. 1999). Toward this end, many researchers have begun using computer vision to sense passive fiducial markings in the scene (Mellor 1995) (Uenohara and Kanade 1995) (Hoff, Lyon et al. 1996) (State, Hirota et al. 1996) (Sharma and Molineros 1997) (Hoff and Vincent 2000). If the positions of the

fiducials are known *a priori*, the pose of the camera with respect to the features can be calculated. However, this requires some preparation or knowledge of the environment.

Recent work by Neumann and You (Neumann and You 1999) used computer vision to detect and track natural features in video images. Point and region features were automatically and adaptively selected for tracking, and their 3-D positions were estimated. The system computed the pose of the camera from the observed features, but did not use an explicit model for the head motion. A single camera was used, which would normally yield results that would have an unknown scale factor in position. However, they used an initial set of features with known positions to compute the absolute pose of the camera over the first set of frames. An EKF was used to compute the 3-D positions of new features, based on the known poses of the camera. Once the 3-D positions of the new features were known sufficiently accurately, they became new fiducials from which to compute camera pose. This allowed the AR system to extend its workspace.

Finally, Azuma, *et al* (Azuma, Lee et al. 1999) developed an AR system for outdoor applications that used gyroscopes in conjunction with computer vision to achieve accurate orientation registration (translation is provided by GPS). The tracked features were assumed to be distant, so that any image motion of the features was presumed due to orientation changes of the head. The observed image motion was mapped into orientation differences and used to provide corrections to inertial sensor drift. A simplified filter was used instead of a true EKF, and there was no motion model.

3 Algorithm Design

To satisfy our goal of accurate and robust head tracking in unstructured environments, we chose to develop a hybrid system consisting of a computer vision system to locate and track

natural features in the scene, as well as inertial sensors (gyroscopes and accelerometers) to provide pose estimates in between vision updates and when vision data is not available. We also chose to incorporate an explicit model of head motion dynamics, to aid in prediction. We designed the algorithm to accommodate more than one head-mounted camera, although it will work with a single camera. We found that two cameras yielded much improved performance over one, as is described later.

We compute both structure (feature point positions) and head pose and motion simultaneously, similar to Broida, *et al* (Broida, Chandrashekhar et al. 1990), since we cannot assume that we have one before the other. But rather than combining all unknowns into a single large state vector, we separate motion estimation and structure estimation into two different EKF's, similar to Ayache, *et al* (Ayache and Faugeras 1989). This reduces computational cost. However, these two filters are coupled together so that each provides information to the other.

This work is novel in that we combine motion and structure estimation, vision and INS sensors, and an explicit head motion model. The system uses no *a priori* information about the scene, and can work in scenes where objects are close or distant. We also examine the consequences of using either one or two cameras as part of the sensor group.

In the subsections below, we first describe the head motion estimation filter. We then describe the structure estimation filter, and then the combined (integrated) system.

3.1 Head Motion Estimation

The frames of reference are shown in Figure 1. The primary frame is the camera frame {C} (which was at the location of right camera in our two-camera apparatus). We assume that the inertial sensors are rigidly mounted with respect to the camera and that the relative pose is between {I} and {C} is known. To simplify the kinematics model of the system below, we set

the {I} frame to have the same orientation as the {C} frame. The world frame {W} is at an arbitrary but fixed location in the scene.

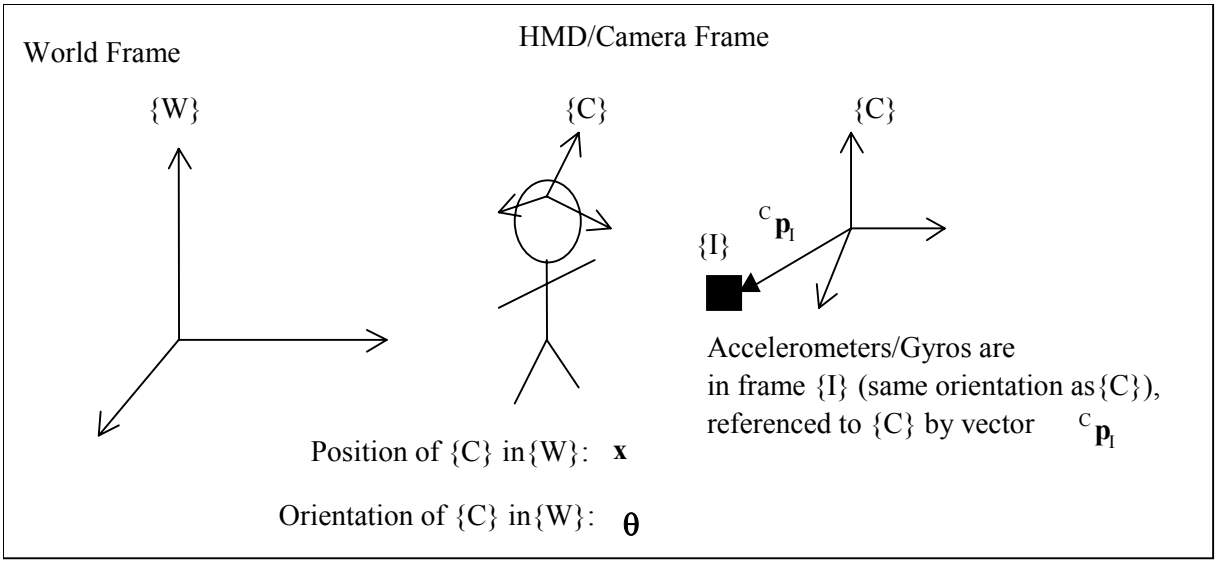


Figure 1 Frames of reference.

Our head motion model assumes constant angular velocity[1] and constant translational acceleration. We represent head motion with a 15x1 state vector $\mathbf{z}_{head} = (\boldsymbol{\theta}, \boldsymbol{\omega}, \mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})^T$ where θ is the orientation of the camera frame with respect to the world (we use Z-Y-X Euler angles in this case, although quaternions can be used with some modifications), ω is the angular velocity, and $\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}$ are the position, velocity, and acceleration of the camera with respect to the world. By combining these into a single state vector, we can represent cross coupling between different axes of the motion, as well as between orientation and translation.

With these states, the discretized system dynamics are as follows (details are in (Chai 2000)):

[1] Empirical data by Zikan, et al (Zikan, Curtis et al. 1994) suggests that this is adequate.

$$\begin{bmatrix} \boldsymbol{\theta}_{k+1} \\ \boldsymbol{\omega}_{k+1} \\ \mathbf{x}_{k+1} \\ \dot{\mathbf{x}}_{k+1} \\ \ddot{\mathbf{x}}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_k + \Delta t \mathbf{W}(\boldsymbol{\theta})\boldsymbol{\omega}_k + \left(\frac{\Delta t^2}{2}\right)\frac{\partial}{\partial \boldsymbol{\theta}}\left[\mathbf{W}(\boldsymbol{\theta})\boldsymbol{\omega}_k\right]\mathbf{W}(\boldsymbol{\theta})\boldsymbol{\omega}_k \\ \boldsymbol{\omega}_k \\ \mathbf{x}_k + \Delta t \dot{\mathbf{x}}_k + \frac{1}{2}\Delta t^2 \ddot{\mathbf{x}}_k \\ \dot{\mathbf{x}}_k + \Delta t \ddot{\mathbf{x}}_k \\ \ddot{\mathbf{x}}_k \end{bmatrix} + \begin{bmatrix} w_k^1 \\ w_k^2 \\ w_k^3 \\ w_k^4 \\ w_k^5 \end{bmatrix} \quad (1)$$

where Δt is the elapsed time since the previous time update, **W** is the inverse of the Jacobian matrix relating the rate of change of the Euler angles to angular velocity, and w_k^i is an unknown input corresponding to the disturbance noise. Notice that with the unknown input equal to zero, the position update equations come from the kinetic relationships between position, velocity, and acceleration with the assumption of linear acceleration between samples. The equation for $\boldsymbol{\theta}_{k+1}$ comes from a Taylor series expansion of $\boldsymbol{\theta}(t)$ where we keep only the first three terms. Thus the model is not a dynamic model in the sense of including mass and inertia, but simply is used to relate measurements that could depend on either positions, velocities, or accelerations. The noise inputs come from the unknown motion of the user (w_k^2, w_k^5), as well as from the linearization error. By grouping signals into vectors in the obvious way, we will use the notation $\mathbf{z}_{k+1} = \mathbf{f}(\mathbf{z}_k) + w_k$. Using our angle set convention, the matrix **W** is given by (details are in (Chai 2000)):

$$\mathbf{W}(\alpha, \beta, \gamma) = \begin{pmatrix} \cos\alpha \sec\beta & \sin\alpha \sec\beta & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ \cos\alpha \tan\beta & \sin\alpha \tan\beta & 1 \end{pmatrix} \quad (2)$$

We have data inputs from three types of sensors: gyroscopes, accelerometers, and cameras. Each type of sensor has associated with it an output equation, which maps the state to the sensor output, **y** = **h**(**z**). The output equations for each sensor are defined as follows.

The gyroscopes produce three angular velocity measurements, one for each axis (units are rad/s), which are related to $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ via (Craig 1990):

$$\mathbf{y}_g = \mathbf{W}^{-1}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{3}$$

The accelerometers produce three acceleration measurements, one for each axis (units are mm/s^2):

$$\mathbf{y}_a = {}^{I}_{W}\mathbf{R}\left(\ddot{\mathbf{x}} + \dot{\boldsymbol{\omega}} \times {}^{W}_{C}\mathbf{R}(\boldsymbol{\theta}) {}^{C}\mathbf{p}_I + \boldsymbol{\omega} \times \frac{d}{dt} {}^{W}_{C}\mathbf{R}(\theta) {}^{C}\mathbf{p}_I + \mathbf{g} \right) \tag{4}$$

where ${}^{W}_{C}\mathbf{R}$ is a matrix rotation from the camera {C} frame to the {W} frame, ${}^{I}_{W}\mathbf{R}$ is the rotation from {W} to {I}, and **g** is gravity.

The computer vision system measures the image position of a target point in the world, in each of the two cameras (units are mm on the image plane). The cameras are separated by a distance *d*, and have their optical axes aligned. Using the estimated world-to-camera pose, we transform the world point into camera coordinates, and then project it onto the images using the perspective projection equations. Here, *f* is the focal length of the camera lenses in mm, ${}^{W}\mathbf{p}_f$ is the position of the feature in the world frame, and ${}^{W}\mathbf{p}_{Corg}$ is the position of the {C} frame origin:

$$\begin{aligned} {}^{C}\mathbf{p}_f &= \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = {}^{C}_{W}\mathbf{R}(\theta)\left({}^{W}\mathbf{p}_f - {}^{W}\mathbf{p}_{Corg} \right) \\ \mathbf{y}_v &= f \begin{bmatrix} p_x / p_z \\ p_y / p_z \\ (p_x + d) / p_z \\ p_y / p_z \end{bmatrix} \end{aligned} \tag{5}$$

In this filter, the feature point positions are considered to be known inputs. In actuality, they are estimates that come from the structure estimation filter, as described below in the next section.

Each sensor has sensor noise associated with it. For example, the camera data measurement at sample time *k* is actually

$$\mathbf{y}_k = \mathbf{y}_v + n_v \tag{6}$$

where n_v is an unknown noise input associated with the computer vision sensor. A similar relation holds for the inertial data. The sensor data varies in the amount of sensor noise, and the rate at which the data arrives. Typically, the data rates of the inertial sensors are much greater than the computer vision.

A statistical description of these unknown inputs and sensor noise in the form of a mean and covariance is used by the extended Kalman filter to determine the appropriate update weightings from the sensor data. That is, we assume that w and n are zero mean white Gaussian sequences with covariance

$$E\left[\begin{bmatrix} w \\ n_g \\ n_a \\ n_v \end{bmatrix}\begin{bmatrix} w \\ n_g \\ n_a \\ n_v \end{bmatrix}^T\right] = \begin{bmatrix} \mathbf{Q}\Delta t & 0 & 0 & 0 \\ 0 & \mathbf{R}_g & 0 & 0 \\ 0 & 0 & \mathbf{R}_a & 0 \\ 0 & 0 & 0 & \mathbf{R}_v \end{bmatrix} \tag{7}$$

The extended Kalman filter (EKF) updates the state estimate $\hat{\mathbf{z}}_k$ and the associated state covariance matrix $\mathbf{P}_k$ as follows (Brown and Hwang 1992). The time update equations are given by:

$$\begin{aligned} \hat{\mathbf{z}}_{k+1}^- &= \mathbf{f}(\hat{\mathbf{z}}_k) \\ \mathbf{P}_{k+1}^- &= \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k + \mathbf{Q}\Delta t \end{aligned} \tag{8}$$

The measurement update equations are given by:

$$\begin{aligned} \mathbf{K} &= \mathbf{P}_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{R})^{-1} \\ \hat{\mathbf{z}}_{k+1} &= \hat{\mathbf{z}}_{k+1}^- + \mathbf{K}(\mathbf{y}_k - \hat{\mathbf{y}}_k) \\ \mathbf{P}_{k+1} &= (\mathbf{I} - \mathbf{K}\mathbf{H}_k)\mathbf{P}_k \end{aligned} \tag{9}$$

where $\mathbf{y}_k$ is the current observation (inertial or camera), $\hat{\mathbf{y}}_k$ is the prediction of the sensor output given the current state estimate, and $\mathbf{A}_k, \mathbf{H}_k$ are the gradient of the dynamics equation and observation equation respectively at the current state estimate.

A block diagram of the head motion estimation filter is shown in Figure 2. We assume that the sensors are asynchronous and their noise is independent of each other, so each sensor can be incorporated using a separate measurement update. The filter will perform the time update step to project the state from the current time step into the next time step, when either gyroscope data, accelerometer data or camera data is available. Then a measurement update step will be followed to update the filter's state according to the new measurement input. This is a recursive process and it will run continuously when the measurement input data is available.

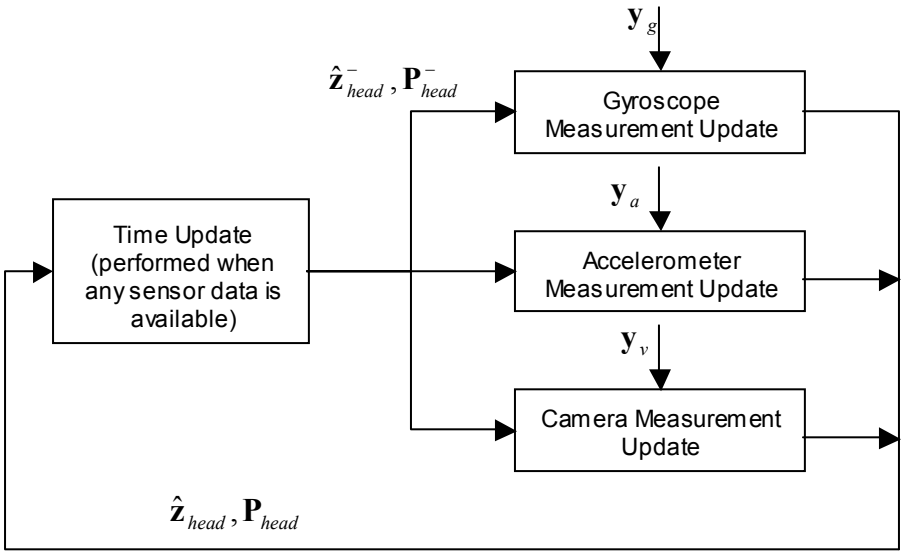


Figure 2 Block diagram of head motion filter.

3.2 Structure Estimation

Scene structure is represented by a single 3Nx1 state vector $\mathbf{z}_{structure} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)^T$ where $\mathbf{x}_i$ is the 3-D position of the i^{th} feature point with respect to the world and N is the number

of points being tracked. In our work we used up to 5 points. We assume that the points are stationary and fixed in the world. We combined all points into a single state vector (similar to Broida, *et al* (Broida, Chandrashekhar et al. 1990)) so that we can represent the cross coupling of the uncertainty between different points.

Note that we represent point positions with respect to the world, even though we do not know the actual location of the world origin. This is not a problem because the world origin is arbitrary, and we are only interested in the relative pose between the user and the scene structure. In actuality, the world origin is determined by the initial guesses of the user's pose and the point positions. An alternative method would be to represent the point positions with respect to the user, at the cost of a more complex time update step. As it is, with the points represented in world coordinates, the time update step is very easy; *i.e.*, $\hat{\mathbf{z}}_{k+1}^{-} = \hat{\mathbf{z}}_k$, $\mathbf{P}_{k+1}^{-} = \mathbf{P}_k + \mathbf{Q}\Delta t$, where $\mathbf{Q}$ has very small values.

Measurement inputs are the observed image feature positions in each of the two cameras (again, in this work we do not address the problem of extracting the feature points and establishing the required correspondences). From each camera, we have N feature points, giving a 2Nx1 measurement vector $\mathbf{y} = (u_1, v_1, \ldots, u_N, v_N)^T$. The sensor measurement equation is $\mathbf{y} = \mathbf{h}(\mathbf{z})$, where $\mathbf{h}$ is a non-linear function and is given by Equation 5. In this filter, which estimates structure, the head pose is considered to be a known input. In actuality, it is an estimate that comes from the motion estimation filter, described above in the previous section. The measurement update equations are given by Equation 9, where $\mathbf{H}$ is a 2Nx2N matrix that is the linearization of $\mathbf{h}$ about the current estimate $\hat{\mathbf{z}}_k$; *i.e.*, $\mathbf{H} = \left[\partial h / \partial z\right]_{\hat{\mathbf{z}}_k}$.

A block diagram of the structure estimation filter is shown in Figure 3. Our implementation allows for the two cameras inputs to be asynchronous, so each sensor can be

incorporated using a separate measurement update. The filter will perform the time update step to project the state from the current time step into the next time step, when data from either camera data, $\mathbf{y}_{left}$ or $\mathbf{y}_{right}$, is available. Then a measurement update step will be followed to update the filter's state according to the new measurement input.

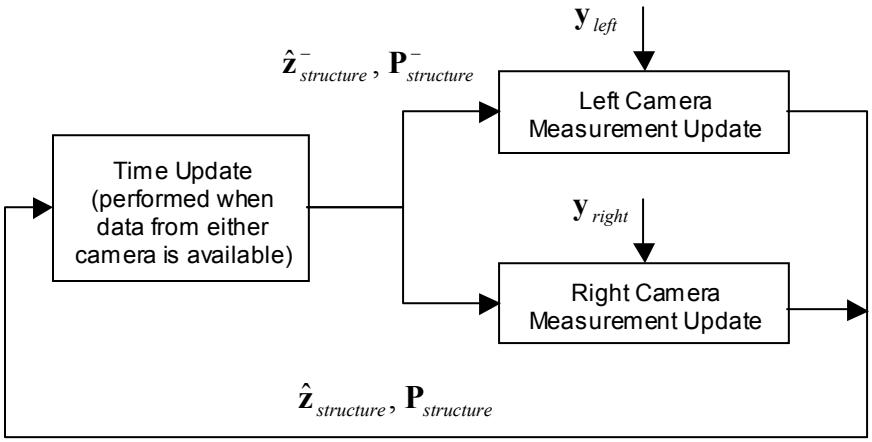


Figure 3 Block diagram of structure estimation filter.

If data from the two cameras are actually available simultaneously, an alternative method is to calculate the 3-D point positions directly using stereo triangulation, and use those as the input data instead of the 2-D image points of each camera. However, sometimes a feature is visible in one camera and not the other, due to occlusion or limited field of view. In that case, our method would still allow data to be used from one camera alone.

3.3 Combined Motion and Structure Estimation

The previous two subsections describe separate Kalman filter algorithms to predict the pose of the user's head and estimate the 3-D structure of the feature points, respectively. To predict the user's head motion, we use inertial sensor data and camera image data and we assume that the true location of the feature points with respect to the world is known. To estimate the 3-

D structure of the feature points, we only use camera image data and we assume that the true camera motion is known.

This subsection describes a combined system that simultaneously predicts the motion and pose of the head as well as the location of the feature points. The idea is to take the results of the head motion prediction filter and use it as a known condition for the estimation of the 3-D structure. Meanwhile, we take the results from the estimation of the 3-D structure filter and use it as a known condition for the head motion prediction step. Thus these two steps form a feedback loop. Figure 4 shows a block diagram of the data flow for the entire process.

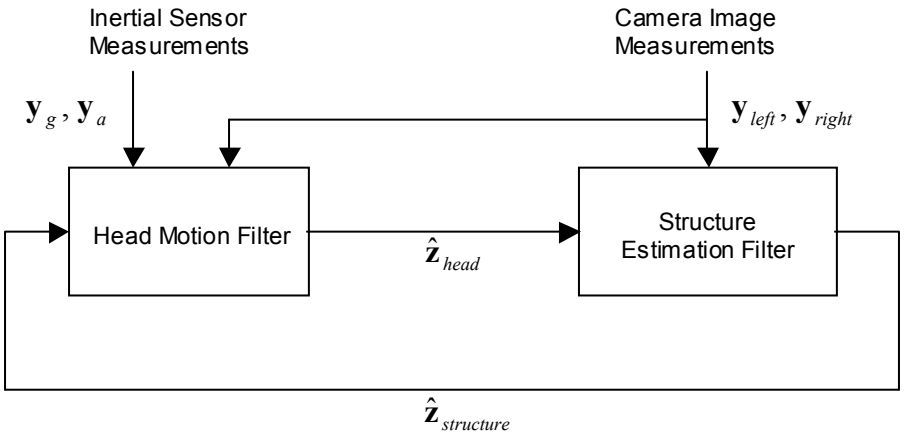


Figure 4 Block diagram of the combined filter algorithm.

The basic structure of the algorithm consists of two EKF's running in parallel. Each filter contains its own time update step and measurement update step. This system is designed to accommodate the fact that camera image data is usually obtained at a slower rate than inertial sensors data. This means that when either gyroscope data or accelerometer data is available, only the head motion prediction filter will perform the time update step and the measurement update step according to the new inertial sensors measurement input. However, when camera

image data is available, both the head motion prediction filter and the feature point prediction filter will perform its own time update step and measurement update step separately.

4 Experimental System

Our experimental system consists of a see-through HMD (Virtual i-o i-glasses) mounted on a helmet (Figure 5). Also attached to the helmet are three small video cameras (Panasonic GP-KS162, with 44-degree field of view). In this work, we used only two of the cameras (left and right). Also attached to the helmet is an inertial sensor system, consisting of a three-axis solid state gyroscope (Watson Industries) and three orthogonally mounted single-axis accelerometers (IC Sensors). Data was transmitted to a desktop computer through a cable tether.

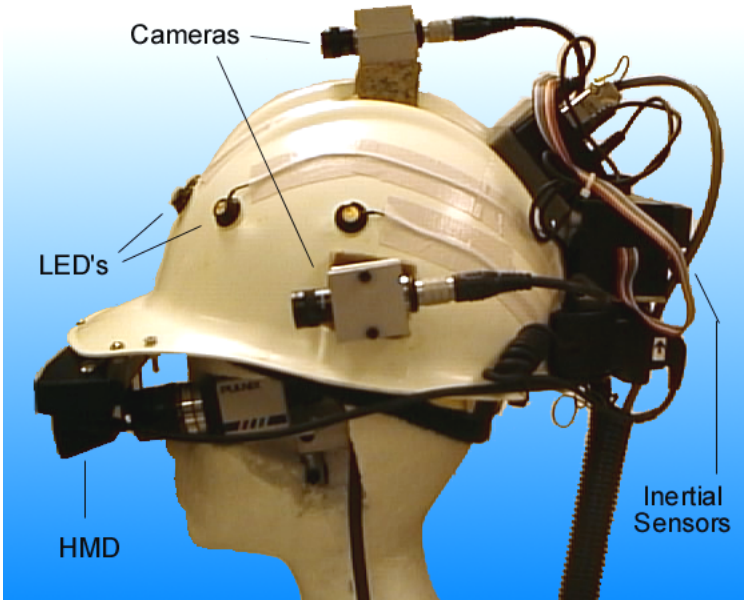


Figure 5 Experimental augmented reality system.

We measured the standard deviation of the noise in the accelerometers to be an average of 36 mm/s^2, and that of the gyroscopes to be approximately 0.1 rad/s (Nguyen 1998). We measured gyroscope biases to be very small compared to the measurement noise, and so ignored them for the remainder of this study.

To obtain ground truth data, we used a Northern Digital Optotrak optical position sensor model 3020, which was mounted on the wall of the lab (Figure 6), and tracked a set of six infrared LED's mounted on the helmet. The Optotrak system uses high-resolution linear array CCD cameras, and can measure the LED positions to an accuracy of 0.15 mm (Rohling, Munger et al. 1995). Since the optical sensor can achieve such a high accuracy, we used it to provide ground truth information of the pose of the user's head, so that we could compare the results of our algorithm to a known result. However, we did not use it as a measurement input for the algorithm that we developed in this research.

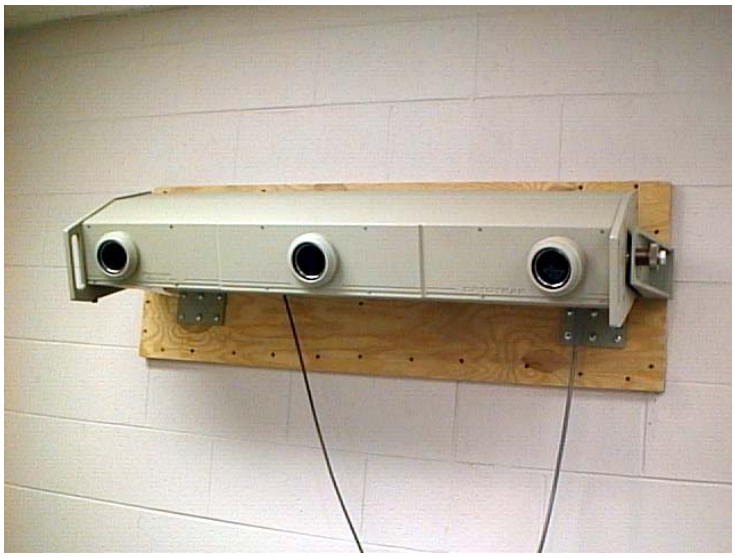


Figure 6 High-accuracy optical sensor used for ground truth.

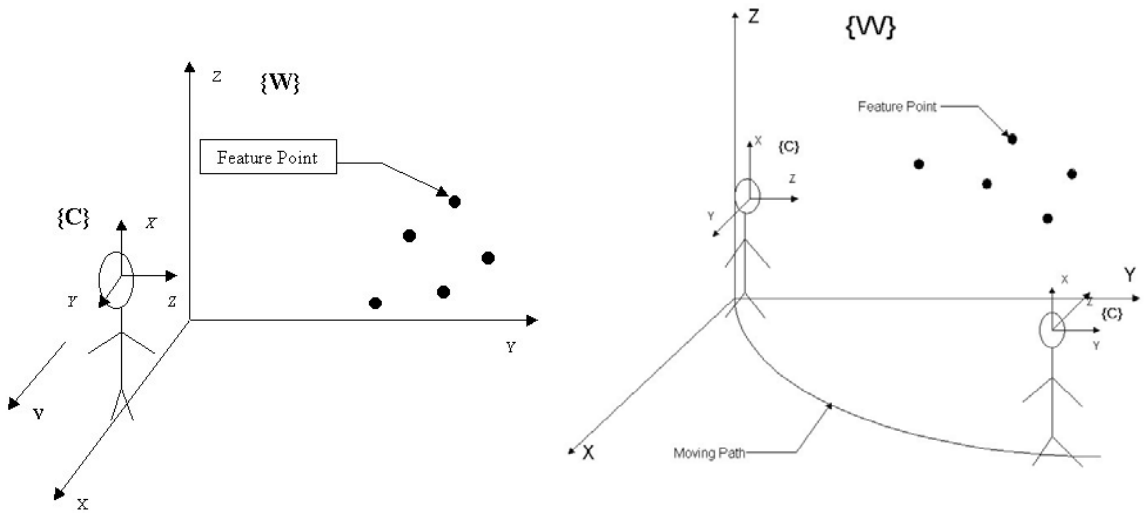
Video images from the head-mounted cameras were digitized to a resolution of 640x480 pixels. We manually processed the images off-line to extract the feature point positions. We estimated the error in feature point measurements to be ± 2 pixels. This is a conservative estimate, and automatic feature extraction algorithms can be much more accurate.

Our current system does not support real-time data acquisition and processing. As described in the next section, we recorded measured data to a file and processed it off-line. The Kalman filter algorithms were written in Matlab and run on a Pentium-class personal computer.

5 Results

5.1 Synthetic Data

We first examined the behavior of the algorithm on purely synthetic data. Four motion trajectories were described, and synthetic accelerometer, gyro, and camera data were generated from them. The first trajectory was a simple translation at constant velocity in the world +X direction, with the user maintaining a fixed gaze parallel to the world +Y axis (Figure 7a). The second trajectory was the same except for a constant acceleration. In the third trajectory, the user moves along a curve with constant velocity in X and constant acceleration in Y, but rotates their head with constant angular velocity in order to keep the feature points in view (Figure 7b).



(a) (b)

Figure 7 Synthetic user motions for trajectories along the +X axis direction (a) or in the XY plane (b).

In the fourth trajectory, an arbitrary motion in the XY plane with varying accelerations was used, but with fixed orientation. A plot of the translation is shown in Figure 8.

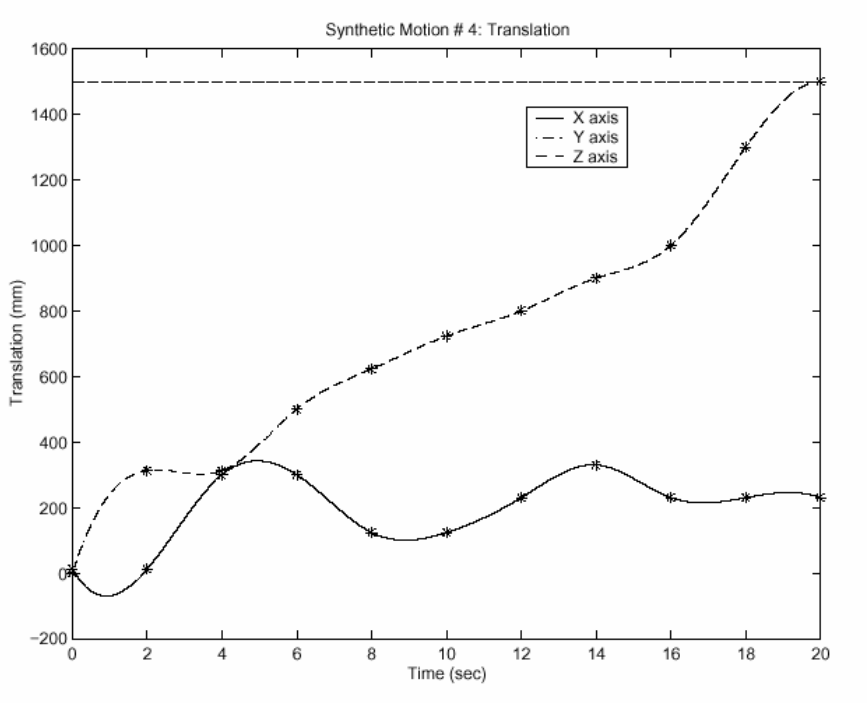


Figure 8 Arbitrary motion in XY plane with varying acceleration, used for synthetic trajectory #4.

Synthetic computer vision data was generated for 5 features, located at points in space around the user. Synthetic white Gaussian noise was added to the measurements with a standard deviation of 0.1 rad/s for gyro data, 36 mm/s^2 for accelerometer data, and 0.04 mm for camera data (corresponding to 2 pixels in the image). The data rates of the sensors were 50 ms for the inertial sensors and 0.5 s for the vision data. The details of these four experiments are summarized in Table 1.

Table 1 Summary of synthetic motion trajectories.

Trajectory	Translation	Gaze direction
1	Constant velocity of 100 mm/s in the +X direction	Fixed in +Y
2	Constant acceleration of 2 mm/s^2 in the +X direction, initial velocity zero	Fixed in +Y
3	Constant velocity of 30 mm/s in +X, constant acceleration of 1.5 mm/s^2 in +Y	Constant angular rotation from +Y to -X

4	Arbitrary motion in XY plane with varying accelerations	Fixed in +Y

Filter parameters were the same for all synthetic experiments. Measurement covariance matrices (R in Equation 9) were set according to the observed values of sensor noise, as described in Section 4. The process noise matrices (Q in Equation 8) were set as follows:

$$\mathbf{Q}_{head} = diag\left(\begin{bmatrix} 10^{-3}\mathbf{I} & \mathbf{I} & 10\mathbf{I} & 10^{6}\mathbf{I} & 10\mathbf{I} \end{bmatrix}\right)$$
$$\mathbf{Q}_{structure} = diag\left(\begin{bmatrix} .02\mathbf{I} & .02\mathbf{I} & .02\mathbf{I} & .02\mathbf{I} & .02\mathbf{I} \end{bmatrix}\right)$$

where *diag* indicates the matrix is block diagonal with the listed elements along the diagonal and **I** is a 3x3 identity matrix. The small values in $\mathbf{Q}_{structure}$ reflect the fact that the point positions are stationary in the world.

For the head motion filter, the values of $\mathbf{Q}_{head}$ were chosen empirically in order to achieve the best performance of the filter. Note that the process noise term for linear velocity is much larger compared to that for the position and the acceleration. Although it is usually difficult to give a simple explanation for the values obtained via tuning, it is clear that the velocity estimate will be modified by measurements very easily. Since we have no direct measurement of velocity, the predicted velocity is derived from past velocity estimates and the acceleration at the current time. Since the inertial sensor suffers from drift over time, the velocity estimate becomes less accurate as well. The large value for velocity process noise covariance will drive the velocity state covariance higher, and when the position term is updated according to the new image measurement, the filter will tend to ignore the current velocity estimate and "reset" the estimate based on the value derived from the position term.

Results are summarized in Table 2. We look at the difference between the predicted and actual point positions, initially and at the end of each run. We measure the 3-D error relative to the user (specifically, the camera frame). However, in many AR applications the accuracy of 2-

D image overlays is more important than actual 3-D scene structure. Thus, we also look at the 2-D projected point positions onto the camera image plane[2]. As one can see, the error decreases significantly from the beginning to the end of each run, in both 3-D and 2-D.

Table 2 Synthetic motion results, showing initial and final feature point errors.

Experiment	Avg. 3-D error (mm)		Avg. 2-D error (mm)	
	Initial	Final	Initial	Final
1	612.23	44.71	0.69	0.07
2	612.23	26.98	0.73	0.04
3	430.08	31.05	0.39	0.04
4	627.07	12.88	0.30	0.03

As is evident in Table 2, initial guesses for 3-D point positions were set fairly large from the true values, to test the algorithm's ability to recover from poor *a priori* knowledge of the scene. In fact, the initial errors are comparable to the distance from the user to the points (about 1 to 2 m). Figure 9 shows the rapid decrease of 3-D point errors for one of the experiments (#3).

[2] We assume that errors in the camera image plane are comparable to those on the HMD display, since the cameras are typically mounted close to the HMD.

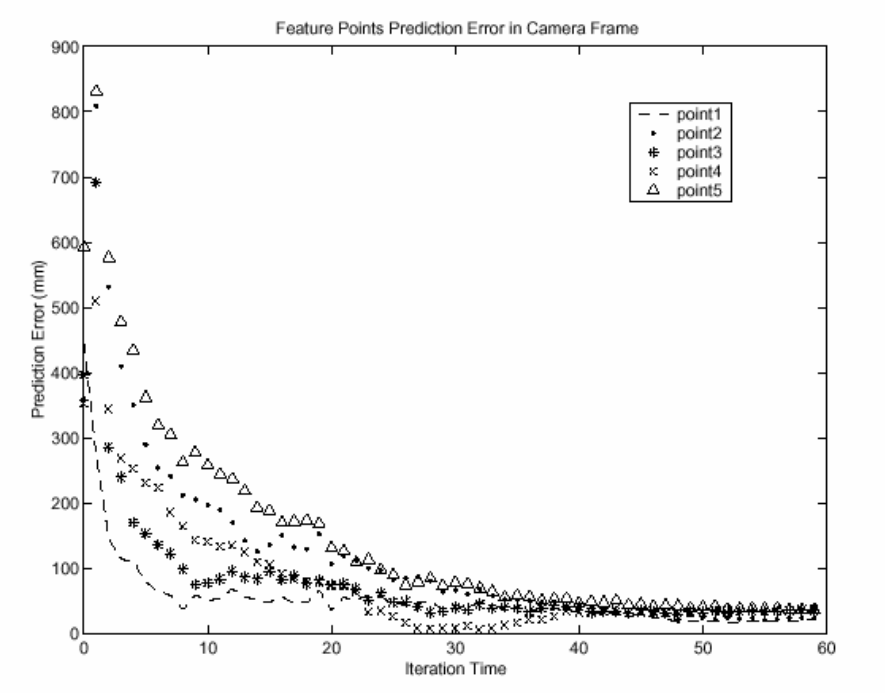
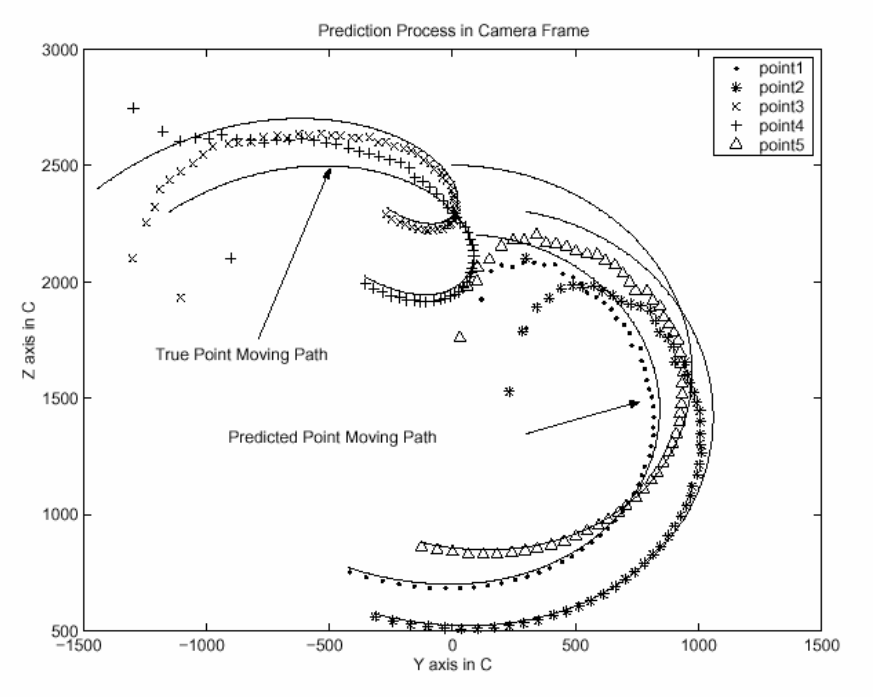
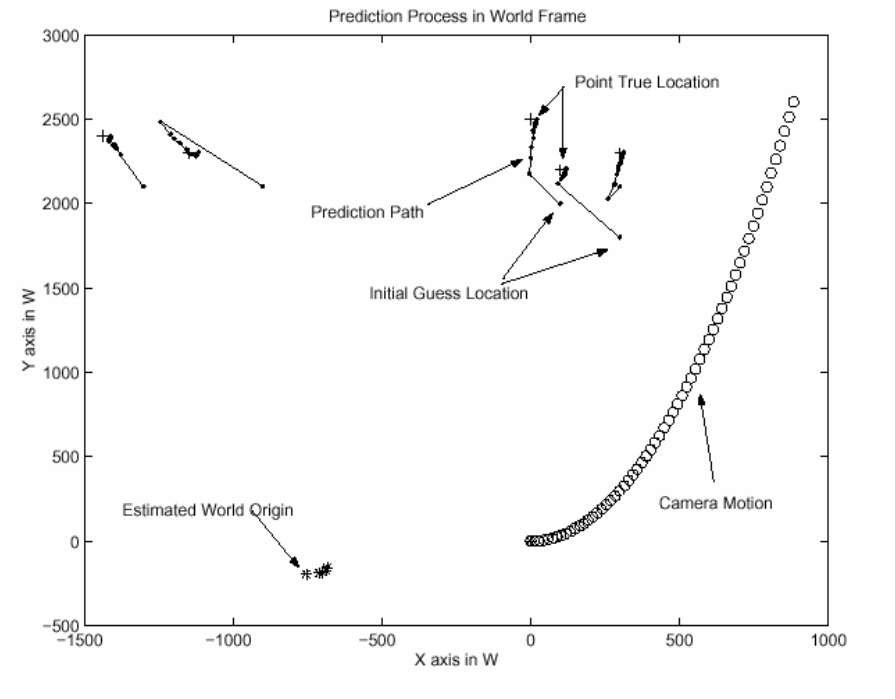


Figure 9 Point prediction error in 3-D, as a function of time (experiment #3).

Figure 10 (a) shows a top view of the prediction process in the {C} frame, also for experiment #3. The solid lines represent the true trajectory of the feature points whereas the markers represent the predicted trajectory for the feature points. The two sets of lines eventually converge at the end of the motion. Figure 10 (b) shows the entire process in the {W} frame viewed from the top. The "o" represents the position of the right camera in the {W} frame, and the "+" represents the true location of feature points. Since the points are fixed in the {W} frame, they do not move during the process. The "." represents the predicted location of feature points. The initial guesses of point locations are fairly far away from the true locations, but move toward the true locations along as time goes on. Note that the estimated world origin also settled down in an arbitrary location.



(a)



(b)

Figure 10 Top view of experiment #3, in {C} frame (a) and in {W} frame (b).

5.2 Real Image Data, Synthetic Inertial Sensor Data

We next tested the algorithm on real image data, but synthetic inertial sensor data. A set of feature points was placed on the surface of a table. We measured the location of the each feature point with respect to the Optotrak sensor mounted on the wall (Figure 11). The helmet (mounted on a tripod) was physically translated from one end of the table to the other. During the movement, the helmet was rotated so that the cameras were aimed at the feature points on the table. The movement was quasi-static, in that we moved the tripod, then collected image data while it was stationary, then moved it again, *etc*. The Optotrak sensor tracked the LED's mounted on the helmet to provide ground truth pose.

Figure 11 Real motion experimental setup, with Optotrak sensor on wall.

We then generated synthetic inertial sensor data corresponding to the real motions recorded by the Optotrak sensor. The ground truth real motion data is shown in Figure 12 and Figure 13.

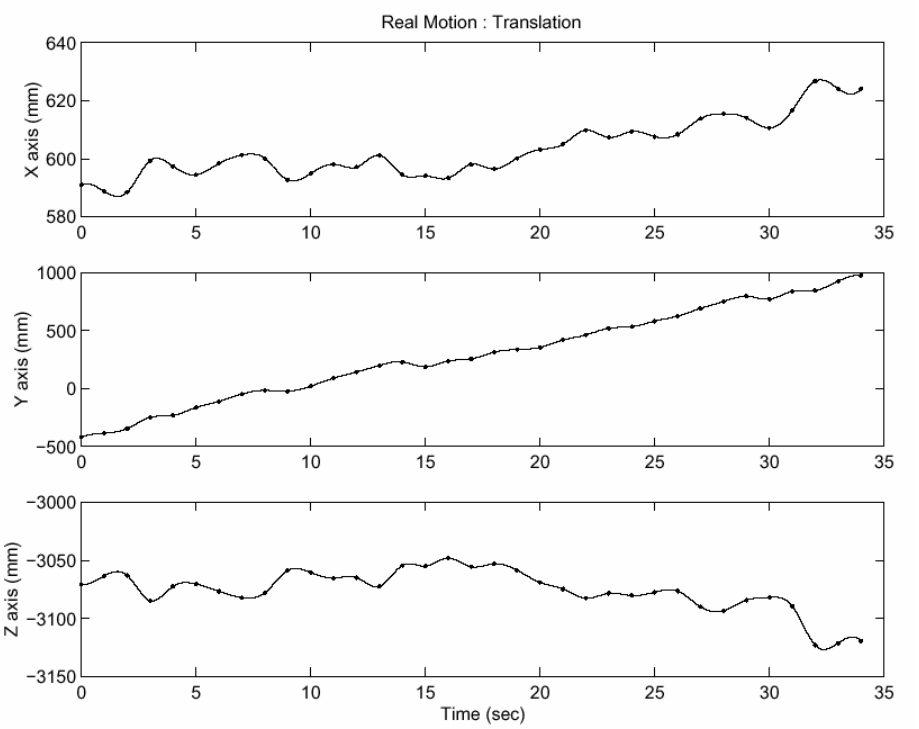


Figure 12 Real motion ground truth translation data.

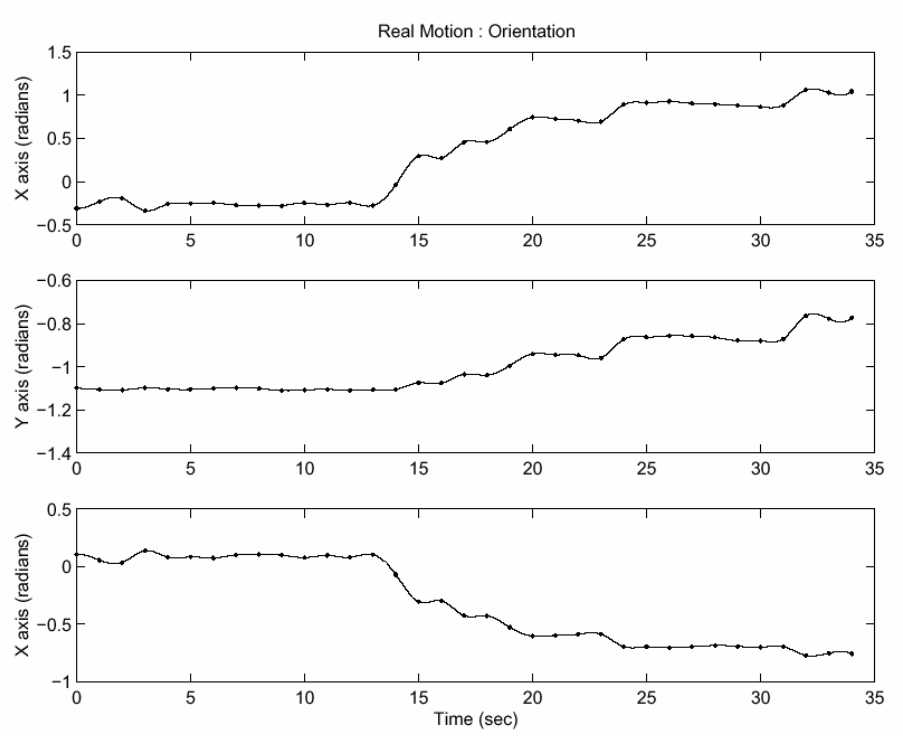


Figure 13 Real motion ground truth orientation data.

Our motion and structure estimation algorithm was then run on this real image data and synthetic inertial sensor data[3]. The result was that the average 3-D point error decreased from an initial value of 1002 mm to a final value of 126 mm at the end of the run, and the average 2-D point error decreased from an initial value of 0.79 mm to a final value of 0.09 mm. Figure 14 shows the decrease of 3-D point errors as a function of time.

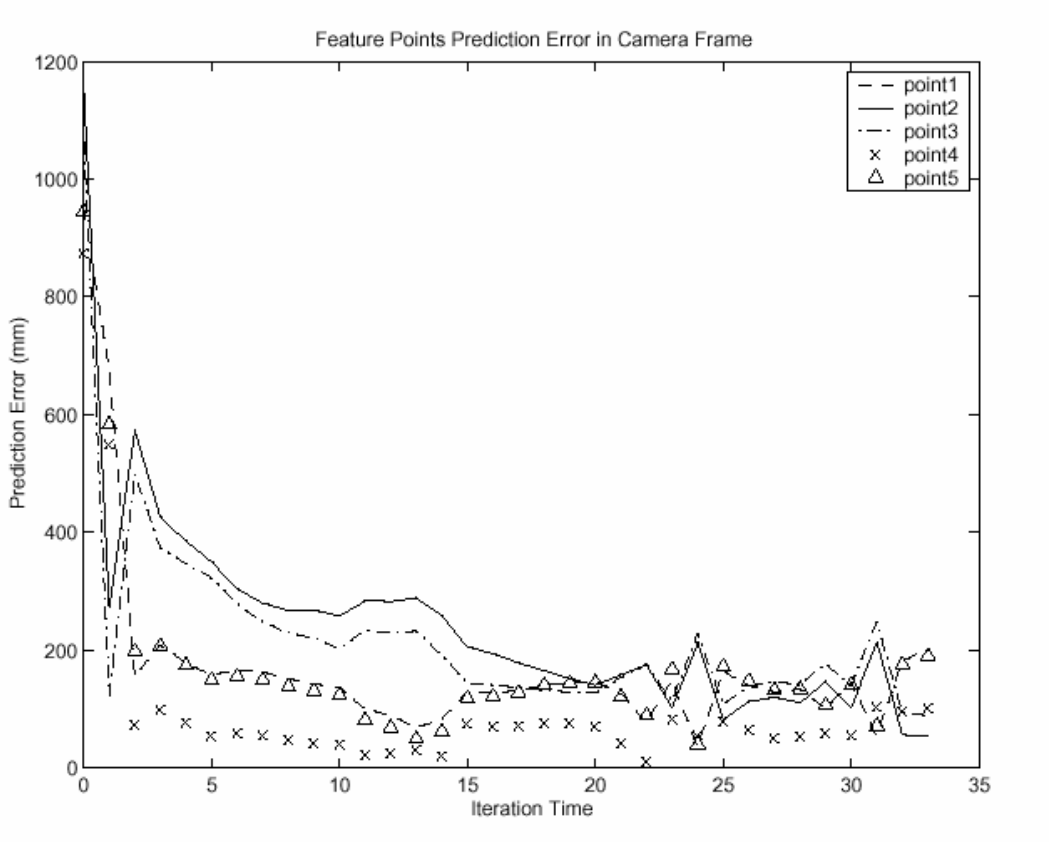


Figure 14 Point prediction error in 3-D, as a function of time (real image data, and synthetic inertial sensor data).

Although the filter converges for real image data, the errors are larger than for synthetic image data. The reason for the larger error is probably due to the crude calibration that was done of the cameras. We only roughly measured the focal length, the camera-to-helmet

[3] Although synthetic, the inertial sensor data is based on measured real motion.

transformation, and the transformation between the Optotrak LED's and the cameras. We also made some simplifying assumptions - that there was no lens distortion in the cameras, and that the optical axes of the two cameras were aligned with each other. All of these calibration error sources could account for the larger error we saw with real data.

5.3 Real Image Data, Real Inertial Sensor Data

We next tested the algorithm on real image data with real inertial sensor data. A person wore the helmet and viewed a calibration object on a table. The calibration object consisted of a pyramid-shaped block covered with a checkerboard pattern (Figure 15). The person executed a series of rotations and translations over a period of approximately 16 seconds, while keeping the block centered in the field of view. During the sequence, we simultaneously recorded digital video from the two head-mounted cameras, as well as inertial sensor data at a rate of 75 Hz.

Figure 15 Experimental setup for tests using real image data and real inertial sensor data.

Inertial sensor data was captured by an A/D board on a PC, running LabView. Simultaneously, an external time signal was also captured and used to "time stamp" the data. This external time signal was generated by a counter, and was continuously shown on a digital display placed in the field of view of the cameras. In this way, we were able to synchronize the video images with the inertial sensor data during the subsequent off line processing.

A set of 100 stereo image pairs were extracted from the digital video streams, at a rate of 6 Hz. We manually identified the time of each image pair by reading the digital display on the counter, visible in each image. Corner features were extracted from each image, using an automatic corner detection algorithm. The locations of 5 corner features that remained visible throughout the move sequence were recorded in each image[4] (Figure 16).

The algorithm was then run on the recorded combined data stream (consisting of image point locations and inertial sensor data), to produce an estimate of head motion and also the 3D locations of the observed feature points. Note that in this test, we did not have ground truth data for the head motion. However, since we did know the 3D structure of the calibration object, we could compare the derived 3D points with the known 3D points on the block.

[4] However, the algorithm can tolerate occasional loss of data. In this run, one or more of the feature points were occasionally obscured in some images.

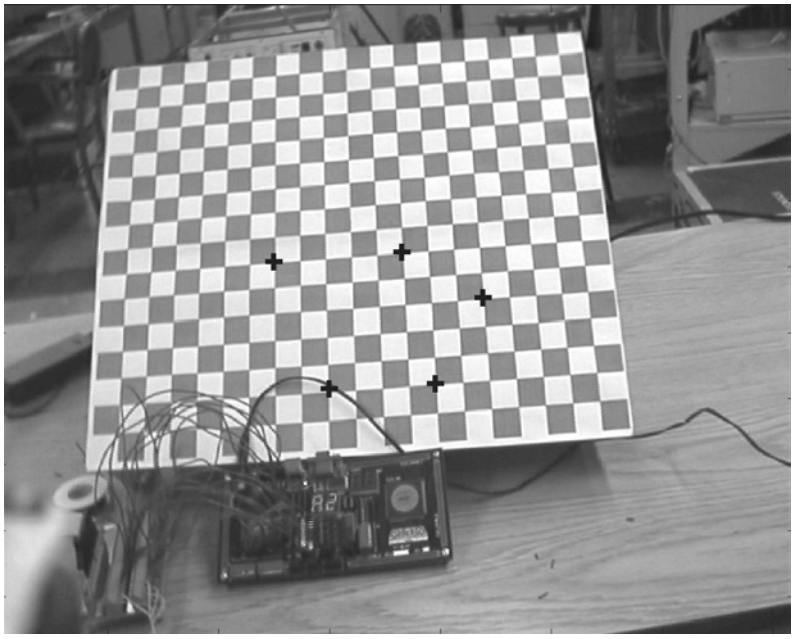


Figure 16 View of calibration object taken from right head-mounted camera, showing the 5 feature points that were tracked (marked by dark crosshairs). Also visible below the checkerboard is the electronic counter used to synchronize the video stream to the inertial sensor data stream.

The algorithm was started with a rough estimate of the locations of the feature points. A 3D depiction of the initial point locations is shown in Figure 17. Note that the points are far from co-planar.

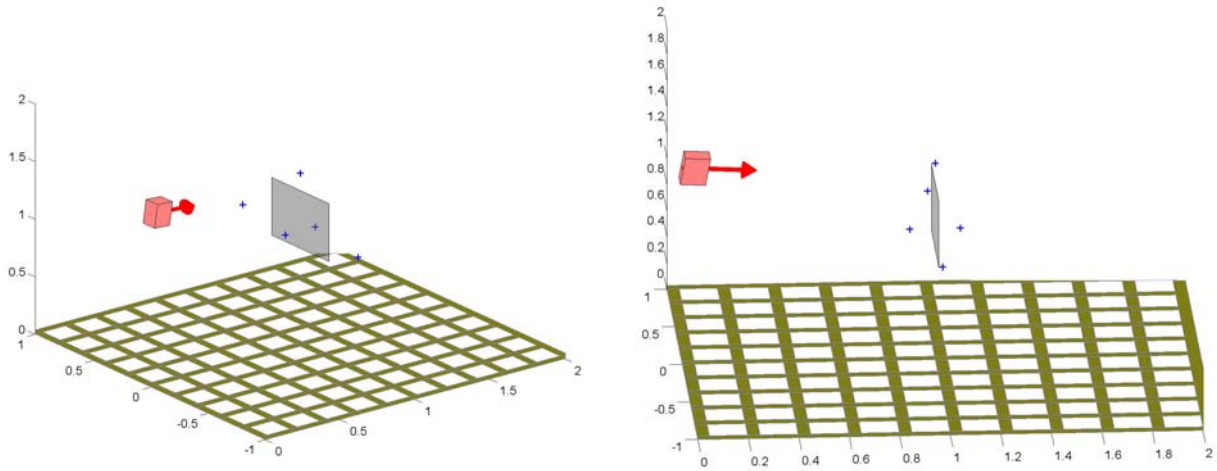


Figure 17 Two views of the initial starting configuration, showing the pose of the head (cube with arrow) and the initial estimates of the point positions.

The final configuration of the camera and the 3D point estimates is depicted in Figure 18. Note that the final point position estimates are almost perfectly coplanar, as they should be since they actually do lie on a plane.

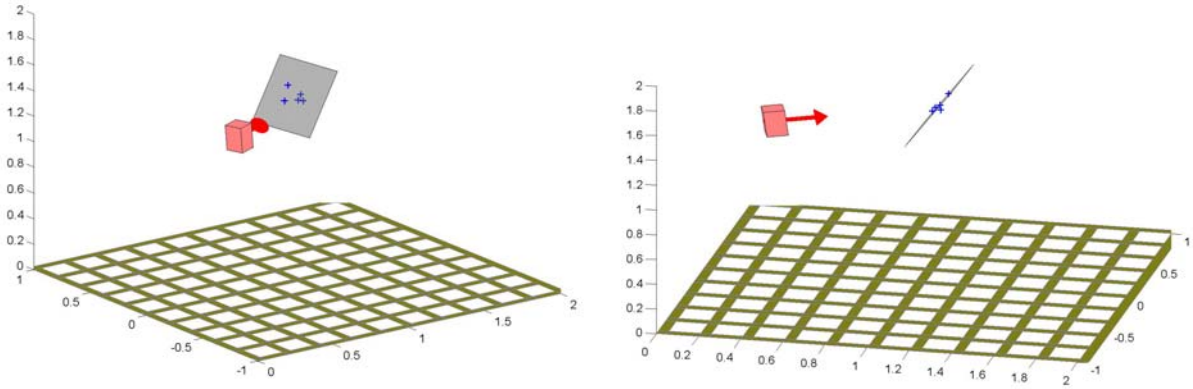


Figure 18 Two views of the final configuration, showing the pose of the head (cube with arrow) and the final estimates of the point positions.

To evaluate the accuracy of the resulting estimated 3D structure, we fit the estimated 3D points to a plane. The RMS (root mean squared) error of the fit of the points to a plane is shown in Figure 19. Initially, the RMS error is over 63 mm, but it rapidly decreases over the motion sequence until it reaches a final value of 10.8 mm. It is possible that these errors could be reduced significantly if better calibration was done of the cameras.

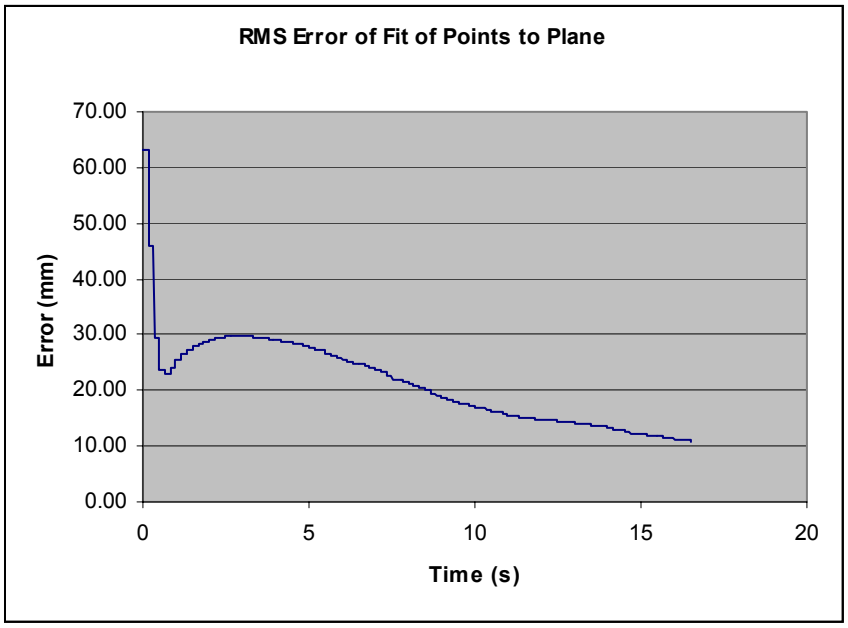


Figure 19 RMS error of the estimated 3-D points, as a function of time (real image data and real inertial sensor data).

5.4 Comparison of One Camera vs. Two Cameras

Our algorithm is designed to be used with any number of cameras, although we used two cameras in the majority of our testing. One issue is whether two cameras are necessary. In principle, a single head-mounted camera with inertial sensor data should be able to recover structure and motion with no unknown scale factor. Effectively, the motion of the single camera provides a baseline separation between two viewpoints, from which triangulation can be done to recover depth unambiguously. However, unlike the stereo case, where the baseline separation is known very accurately, in the single camera case we only have a rough estimate of the motion. As a result, we would expect structure estimation to be less accurate for a single camera system than for a two camera system.

We performed a test to see if this was true, and to see if more accurate inertial sensor data could compensate for the loss of one camera. A single camera system was run on the synthetic

data from experiment 4 (Figure 8). The amount of noise added to the inertial sensor data was varied, in 5 different trials. The results are shown in Table 3. The top row of the table shows the results for the two camera system, previously shown in Table 2, and is repeated here for convenience. The remaining rows of the table show the results for the one camera system, with varying levels of noise for the accelerometers and gyros.

Table 3 Comparison of structure estimation errors for one camera vs. two cameras.

	σ^2 (accell), $(mm/s^2)^2$	σ^2 (gyros), $(rad/s)^2$	Avg initial 3D error (mm)	Avg final 3D error (mm)	Avg initial 2-D error (mm)	Avg final 2-D error (mm)
Two cameras	1300	0.01	627.07	12.88	0.30	0.03
One camera	1300	0.01	627.07	1066.17	0.34	0.34
	130	0.001	627.07	461.29	0.35	0.16
	13	0.0001	627.07	198.65	0.34	0.06
	1.3	0.00001	627.07	65.15	0.34	0.13
	0.13	0.000001	627.07	51.53	0.36	0.09

The tests confirm that the single camera system does not perform as well as the two camera system, in terms of the final 2-D and 3-D prediction errors in the point positions. When the variance of the inertial sensor noise is decreased by three orders of magnitude, the final prediction errors appear to level off, although they are still much higher than the errors in the two camera system. We conclude that two cameras may be required for practical tracking in AR. This is a possible area of future investigation.

6 Discussion

We have developed a new method for AR registration that simultaneously tracks the pose and motion of the user's head, as well as estimating the 3-D locations of naturally occurring

features in the scene. Relying only on head-mounted cameras and inertial sensors, the method is applicable to portable systems, and in completely unstructured and unprepared environments. Unlike previous work in this area, this method requires no *a priori* knowledge about the scene, and can work in environments where the objects of interest are close to the user. We experimentally found good convergence on testing with a combination of real and synthetic data. Although our method can work with any number of cameras, we found that two cameras gave much better performance than a single camera.

Our method assumes that visual features exist in the scene, that they persist over time, and are stationary. Our work primarily addressed the algorithmic issues of how to fuse data from (possibly asynchronous) camera and inertial data sensors, in order to estimate structure and motion. We did not address many other difficult problems, such as detecting and tracking features, calibration, and real time operation. Some progress has been made on these issues by other researchers, and they remain an active area of research.

A possible area of future investigation is how to tune the filter parameters (specifically, the process noise covariance matrices) to achieve optimum performance. Ideally, instead of manually adjusting the parameters based on an intuitive understanding of the process, an automatic procedure would be used such that optimum performance of the system is guaranteed. One interesting approach is to adaptively adjust the filter parameters based the observed motion of the user. Intuitively, if the person is moving slowly and smoothly, we effectively have a lower level of process noise (*i.e.*, unmodeled disturbances) than if the person is moving rapidly and accelerating quickly, *etc.* We have done some preliminary work in this area, in developing an adaptive estimation approach based on a multiple model estimator (Chai, Nguyen et al. 1999).

7 Acknowledgements

We would like to acknowledge the valuable contribution made by Khoi Nguyen, who developed the equations for head dynamics and much of the Matlab software to implement the Kalman filter. The experimental augmented reality system was built by Tory Lyon, Khoi Nguyen, and Rex Rideout. Northern Digital, Inc., provided valuable assistance to us in using the Optotrak sensor. Tao Xiong performed most of the experimental work with real image data and real inertial sensor data.

8 References

Ayache, N. and O. Faugeras (1989). "Maintaining representations of the environment of a mobile robot." IEEE Trans. on Robotics and Automation **5**(6): 804-819.

Ayache, N. and O. D. Faugeras (1988). "Building, Registrating, and Fusing Noisy Visual Maps." International Journal of Robotics Research **7**(6): 45-65.

Azarbayejani, A. and A. P. Pentland (1995). "Recursive estimation of motion, structure, and focal length." IEEE Trans Pattern Anal Mach Intell **17**(6): 562-575.

Azuma, R. and G. Bishop (1994). Improving static and dynamic registration in an optical see-through HMD. 21st International SIGGRAPH Conference, Orlando, FL, USA, ACM; New York NY USA.

Azuma, R., B. Hoff, et al. (1999). A motion-stabilized outdoor augmented reality system. IEEE Virtual Reality '99, IEEE.

Azuma, R., J. W. Lee, et al. (1999). "Tracking in unprepared environments for augmented reality systems." Computers and Graphics (Pergamon) **23**(6): 787-793.

Azuma, R. T. (1997). "A Survey of Augmented Reality." Presence **6**(4): 355-385.

Barshan, B. and H. F. Durrant-Whyte (1995). "Inertial Navigation Systems for Mobile Robots." IEEE Transactions on Robotics and Automation **11**(3): 328-342.

Bhanu, B., S. Das, et al. (1996). "System for obstacle detection during rotorcraft low altitude flight." IEEE Trans Aerosp Electron Syst **32**(3): 875-897.

Borenstein, J., H. R. Everett, et al. (1996). Navigating Mobile Robots: Systems and Techniques. Wellesley, MA, A. K. Peters, Ltd.

Broida, T. J., S. Chandrashekhar, et al. (1990). "Recursive 3-D motion estimation from a monocular image sequence." IEEE Trans Aerosp Electron Syst **26**(4): 639-656.

Brown, R. G. and P. Y. C. Hwang (1992). Introduction to random signals and applied Kalman filtering. New York, J. Wiley.

Chai, L. (2000). 3-D Motion and Structure Estimation Using Inertial Sensors and Computer Vision for Augmented Reality. Engineering Division. Golden, CO, Colorado School of Mines**:** 110.

Chai, L., K. Nguyen, et al. (1999). An adaptive estimator for registration in augmented reality. 2nd Int'l Workshop on Augmented Reality, San Franscisco, IEEE.

Craig, J. (1990). Introduction to Robotics: Mechanics and Control. Reading, Massachusetts, Addison Wesley.

Crowley, J. L. (1996). Mathematical Foundations of Navigation and Perception for an Autonomous Mobile Robot. Reasoning with Uncertainty in Robotics. L. Dorst, M. van Lambalgen and F. Voorbraak. New York, Springer-Verlag**:** 9-51.

Debrunner, C. H. and N. Ahuja (1998). "Segmentation and factorization-based motion and structure estimation for long image sequences." IEEE Trans Pattern Anal Mach Intell **20**(2): 206-211.

Dickmanns, E. D. and B. D. Mysliwetz (1992). "Recursive 3-D Road and Relative Ego-State Recognition." IEEE Trans. Pattern Analysis and Machine Intelligence **14**(2): 199-213.

Fang, J. Q. and T. S. Huang (1984). "Some experiments on estimating the 3-d motion parameters from a sequence of image frames." IEEE Trans. on Pattern Analysis and Machine Intelligence **PAMI-6**(5): 545-554.

Faugeras, O. D. (1993). Three-dimensional computer vision - a geometric viewpoint. Cambridge, Massachusetts, MIT Press.

Forsberg, J., U. Larsson, et al. (1995). "Mobile robot navigation using the range-weighted Hough transform." IEEE Robotics & Automation Magazine **2**(1): 18-26.

Foxlin, E. (1996). Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter. VRAIS, Santa Clara, California, IEEE Computer Society.

Foxlin, E., M. Harrington, et al. (1998). Constellation: A Wide-range Wireless Motion-tracking System for Augmented Reality and Virtual Set Applications. SIGGRAPH 1998, Orlando, FL.

Hoff, W. and C. Sklair (1990). Planetary Terminal Descent Hazard Avoidance Using Optical Flow. IEEE Conf. Robotics and Automation.

Hoff, W. A., T. Lyon, et al. (1996). Computer Vision-Based Registration Techniques for Augmented Reality. Intelligent Robots and Computer Vision XV, Boston, Massachusetts, SPIE.

Hoff, W. A. and T. Vincent (2000). "Analysis of Head Pose Accuracy in Augmented Reality." IEEE Trans. Visualization and Computer Graphics.

Kriegman, D. J., E. Triendl, et al. (1989). "Stereo vision and navigation in buildings for mobile robots." IEEE Transactions on Robotics and Automation **5**(6): 792-803.

Matthies, L. and S. A. Shafer (1987). "Error modeling in stereo navigation." IEEE Journal of Robotics and Automation **RA-3**(3): 239-248.

Mellor, J. P. (1995). Realtime camera calibration for enhanced reality visualization. Computer Vision, Virtual Reality and Robotics in Medicine. First International Conference, CVR Med'95, Nice, France, Springer-Verlag; Berlin Germany.

Meyer, K., et al (1992). "A Survey of Position Trackers." Presence **1**(2): 173-200.

Moravec, H. P. (1988). Sensor Fusion in Certainty Grids for Mobile Robots. AI Magazine**:** 61-74.

Morita, T. and T. Kanade (1997). "Sequential factorization method for recovering shape and motion from image streams." IEEE Trans Pattern Anal Mach Intell **19**(8): 858-867.

Neumann, U. and S. You (1999). "Natural Feature Tracking for Augmented Reality." IEEE Transactions on Multimedia **1**(1): 53-64.

Nguyen, K. (1998). Inertial Data Fusion Using Kalman Filter Methods for Augmented Reality. Engineering Division. Golden, Colorado, Colorado School of Mines**:** 259.

Rohling, R., P. Munger, et al. (1995). "Comparison of Relative Accuracy Between a Mechanical and an Optical Position Tracker for Image-Guided Neurosurgery." Journal of Image Guided Surgery **1**: 30-34.

Sammarco, J. J. (1994). A Navigational System for Continuous Mining Machines. Sensors**:** 11-17.

Sharma, R. and J. Molineros (1997). "Computer vision-based augmented reality for guiding manual assembly." Presence **6**(3): 292-317.

Shekhar, C. and R. Chellappa (1992). "Passive ranging using a moving camera." Journal of Robotic Systems **9**(6): 729-52.

Soatto, S. and P. Perona (1998). "Reducing `structure from motion': A general framework for dynamic vision Part 2: Implementation and experimental assessment." IEEE Trans Pattern Anal Mach Intell **20**(9): 943-960.

State, A., G. Hirota, et al. (1996). Superior augmented reality registration by integrating landmark tracking and magnetic tracking. 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96), New Orleans, LA, USA, ACM; New York NY USA.

Tomasi, C. and T. Kanade (1992). "Shape and motion from image streams under orthography: A factorization method." Int J Comput Vision **9**(2): 137-154.

Uenohara, M. and T. Kanade (1995). "Vision-based object registration for real-time image overlay." Computers in Biology and Medicine **25**(2): 249-60.

Weiß, G., C. Wetzler, et al. (1994). Keeping Track of Position and Orientation of Moving Indoor Systems by Correlation of Range-Finder Scans. 1994 International Conference on Intelligent Robots and Systems (IROS'94), Munich, Germany.

You, S., U. Neumann, et al. (1999). Hybrid inertial and vision tracking for augmented reality registration. IEEE Virtual Reality '99, IEEE.

Zikan, K., W. D. Curtis, et al. (1994). A note on dynamics of human head motions and on predictive filtering of head-set orientations. Telemanipulator and Telepresence Technologies, SPIE.

9 Figure Captions

Figure 1 Frames of reference.

Figure 2 Block diagram of head motion filter.

Figure 3 Block diagram of structure estimation filter.

Figure 4 Block diagram of the combined filter algorithm.

Figure 5 Experimental augmented reality system.

Figure 6 High-accuracy optical sensor used for ground truth.

Figure 7 Synthetic user motions for trajectories along the +X axis direction (a) or in the XY plane (b).

Figure 8 Arbitrary motion in XY plane with varying acceleration, used for synthetic trajectory #4.

Figure 9 Point prediction error in 3-D, as a function of time (experiment #3).

Figure 10 Top view of experiment #3, in {C} frame (a) and in {W} frame (b).

Figure 11 Real motion experimental setup, with Optotrak sensor on wall.

Figure 12 Real motion ground truth translation data.

Figure 13 Real motion ground truth orientation data.

Figure 14 Point prediction error in 3-D, as a function of time (real data).