Robust Visual Tracking for Non-Instrumented Augmented Reality

Georg Klein and Tom Drummond
Department of Engineering
University of Cambridge
Cambridge CB1 2PZ, UK
{gswk2,twd20}@eng.cam.ac.uk

Abstract

This paper presents a robust and flexible framework for augmented reality which does not require instrumenting either the environment or the workpiece. A model-based visual tracking system is combined with with rate gyroscopes to produce a system which can track the rapid camera rotations generated by a head-mounted camera, even if images are substantially degraded by motion blur. This tracking yields estimates of head position at video field rate (50Hz) which are used to align computer-generated graphics on an optical see-through display. Nonlinear optimisation is used for the calibration of display parameters which include a model of optical distortion. Rendered visuals are pre-distorted to correct the optical distortion of the display.

1. Introduction

Augmented Reality (AR) is the synthesis of real and virtual imagery. Computer-generated images are added to a user's view of a real environment to provide extra information and to enhance the user's perception of this environment. For example, augmented reality may add virtual explanatory labels to real objects, or enable the user to see objects which do not exist (e.g. a planned building extension) or which are normally hidden from view (x-ray vision.)

If virtual imagery is to make sense when superimposed on a real background, these images must be exactly aligned with the real world: a real chess-board with virtual pieces drawn a few centimeters out of alignment, for example, is useless. The accurate alignment of real and virtual images is known as *registration* and remains challenging to this day.

In the context of augmented reality using a head-mounted display (HMD), a common approach to registration is to mount a sensor on the user's head. Assuming the position of the sensor relative to the display is fixed, registration may be split into two parts: an accurate *calibration* of each eye's display relative to the sensor, and robust and accurate *tracking* of the sensor's position in the world.

A wide variety of tracking technologies have been applied to AR and are described in recent surveys [2, 21]. Magnetic and ultrasound sensors have been used successfully, but confine the user to an instrumented working volume (which may be very small.) Video cameras on the other hand can operate without external beacons, and video capture capability is becoming a standard feature in off-the-shelf PCs. Consequently, the use of video cameras as sensors for tracking has been the subject of substantial research. In particular, *visual servoing* techniques originating from robotics have recently been applied to AR tracking [18, 23, 4]. These systems measure errors between the camera video feed and a re-projection of the scene based on the estimated camera pose, and update the pose estimate to minimise these errors.

By contrast to robotics applications where motions may be controllable and predictable, tracking a user's head motions with a head-mounted camera is challenging. Rapid head rotation causes image features to undergo large motion between frames which, if not expected, can cause visual tracking systems to fail. This has led to the development of hybrid tracking systems which utilise *sensor fusion* to combine a camera with another form of sensor. Inertial sensors are a popular choice [29, 20] due to their good high-frequency response and independence from external beacons and these sensors are used here.

This paper presents a head-mounted, optical see-through AR system based on hybrid tracking. A visual servoing system originally developed for the control of a welding robot [6] has been adapted for head-mounted operation. This system relies on a CAD model of edges occurring in the scene: thus, the system is independent of any external beacons and does not require the environment to be fitted with markers. The system has been adapted to operate at 50Hz and supports wide-angle lenses with on-line radial distortion calibration. The addition of inertial sensors has made the system capable of tracking rapid rotations, even in the face of substantial image degradation due to motion blur [14].

Further, this paper demonstrates that the visual servoing techniques employed for tracking may also be used for sensor-to-eye calibration of the optical see-through HMD. An eleven-parameter model is fitted for each eye. This model includes a radial distortion term which is used to correctively pre-distort the augmented visuals.

Section 2 of this paper gives an overview of previous work in the field. Section 3 describes the head tracking strategy employed. Section 4 shows the application of visual servoing to the HMD calibration. The performance of the system is presented in in Section 5.

2 Background

One of the earlier implementations of visual servoing in robotics may be found in [7]. Espiau and Chaumette use visual servoing to guide a six-jointed robot to a target position. At each time step, a Jacobian matrix relating the robot's degrees of freedom to the image error between current and target position is calculated, and the robot joint velocities set so as to minimise the image error. The same procedure (minus the robot) has been used to track camera pose in several augmented reality systems [18, 23, 4] which differ in type of image feature employed for measurement. Sundareswaran and Behringer [23] use circular concentric ring fiducial markers which are placed at known locations on a computer case to overlay the hidden innards of the case on the camera's video feed. These markers are unique and can be found by searching the image; this allows the system to initialise itself from an unknown viewpoint and prevents tracking failure due to correspondence errors, but incurs a computational performance penalty. Behringer et al. [4] extend this approach to use edges and corners already present in the scene. These features are described in a CAD model; this is the approach also used here. Marchand and Chaumette [18] demonstrate visual servoing with a number of features, including points, lines and (the edges of) cylinders, all of which have known 3D locations.

Other visual tracking approaches have attempted to use features in the scene for which no prior CAD model exists. Genc et al. [8] use a structure-from-motion approach to learn the position of trackable natural features in a scene. During this learning stage, a marker-based tracking system is used to determine camera pose. Once suitable features - typically corners or texture highlights - have been selected, the markers are no longer necessary. Park et al. [19] use a similar approach to extend the useful range of a fiducial-based tracking system, however the fiducials are not removed once new features have been learned. Kanbara et al. [13] similarly use stereo vision to extract the natural feature positions, which are used to extend tracking range.

Some visual tracking systems used for augmented reality do not depend on information from previous frames and instead calculate a pose estimate from scratch every frame [11]. Many other systems, including the system presented here, require information from a previous state to calculate camera pose at the current frame. Such systems often work on the assumption that motion between frames will be relatively small - this premise allows first-order differentials to be used and greatly simplifies the correspondence problem. However, such systems by themselves are often not capable of tracking the rapid image motions which may occur with head-mounted cameras. It is common for AR systems to combine visual tracking with an alternative sensor by *sensor fusion*. This is frequently done using a statistical filter framework, e.g. by using an extended Kalman filter (EKF). Azuma and Bishop [3] combine LED beacon tracking with inertial sensors in an EKF, and use pose prediction to improve dynamic registration with an optical see-through HMD. The problem of obtaining a suitable probability model for use in the EKF is tackled by Chai et al. in [5]. Fiducial tracking and inertial sensors are combined by You and Neumann [29]. Yokokohji et al. [28] go to great lengths to reduce the system latency and dynamic registration error of a combined fiducial and inertial tracking system. The fusion of magnetic sensors and visual tracking has also been applied to AR [22, 1]; in contrast to inertial sensors, magnetic sensors measure position directly and can therefore prevent a complete loss of tracking. Finally, Koller et al. [15] demonstrate that an EKF can also be used to filter the measurements of a stand-alone visual tracker.

The calibration of head-mounted displays for AR has also been the focus of substantial research. When video see-through displays are used, a large number of measurements can be made from the video stream and standard computer vision camera calibration techniques can be employed. Tsai [24] introduces a rich camera model including radial distortion and proposes a two-stage calibration technique, consisting of an initial linear solution followed by a nonlinear optimisation. Marchand and Chaumette [17] use a nonlinear optimisation using hundreds of measurements made across many frames of a video sequence to calibrate a similar model. When optical see-through displays are used, measurements can no longer be made automatically but must be supplied by the user. Ergonomics limit the number and accuracy of measurements a user can be expected to make, and hence the camera models used are generally more simple. Azuma and Bishop [3] use a four-step calibration procedure in which the sensor-to-eye transformation is found be having the user align his or her eye with two bore-sights. Lens distortion is not modeled. Tuceryan and Navab [25] do not separate the calibration into separate intrinsic and extrinsic parameters and instead calibrate a 3×4 11-DOF projection matrix in a single linear step. Measurements are gathered using a single 3D point with known coordinates. The user sees a cross-hair in the display and aligns this with the 3D point by moving his or her head. Twelve measurements are made with the cross-hair in various positions. Genc extends this system to stereo by replacing the monocular cross-hairs with a 3D disc marker in [9]. Neither of these systems model radial distortion.

The tracking system presented here works on a principle similar to Harris' RAPiD system [10]. RAPiD tracks objects such as fighter aircraft using a CAD model of object edges. The error between the actual object and a re-projection is measured by small one-dimensional edge searches performed from control points on the CAD model edges. In contrast to other model-edge based tracking approaches [16], this local search avoids the computational burden of full-frame edge-detection and allows 50Hz real-time operation. This approach is extended by Drummond and Cipolla in [6]. A BSP-tree based strategy replaces RAPiD's view-sphere to determine control point visibility. Further, an on-line camera calibration is added to the system, which is applied to the servoing of a welding robot. The addition of inertial sensors and a sensor fusion strategy to this system are described in [14].

3 Head Tracking

This section describes the tracking system employed. After an introductory description of the underlying visual servoing principle, details of the visual tracking system's operation are described.

3.1 Visual Tracking

In general, image-based visual *servoing* of a robot attempts to control a robot in such a way that an image measurement of the error between the current robot position and a target position is minimised. In visual *tracking*, an estimate of camera pose takes the place of the robot. The camera pose estimate is updated in such a way as to minimise the error between the camera's real view of the world and a projection of the world generated from the current pose estimate.

The control strategy for tracking, which is illustrated in Figure 1, is as follows:

- Step 1: A video image is acquired from the video capture hardware.
- Step 2: A projection of the world from the current pose estimate is rendered.
- Step 3: A vector $\boldsymbol{e}$ of image errors between the projection and the video image is measured.
- Step 4: The pose estimate is updated by a vector of motion parameters $\boldsymbol{\mu}$ to minimise these image errors.

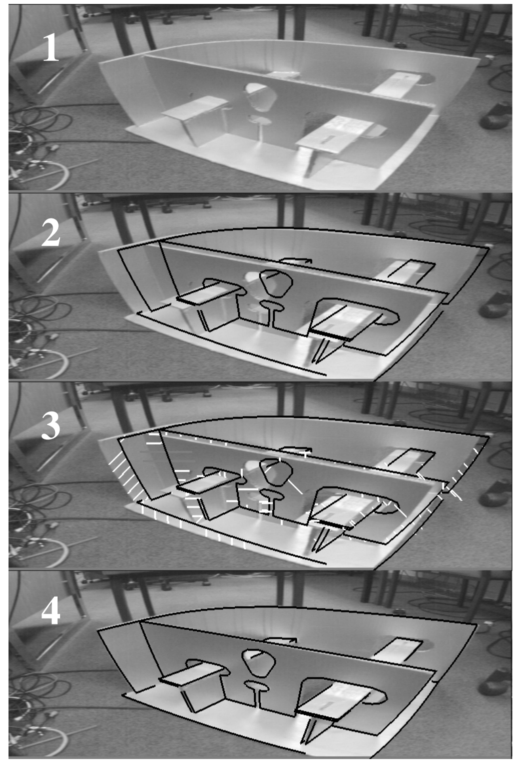

Figure 1. Tracking system loop

To calculate $\boldsymbol{\mu}$, a Jacobian matrix J which describes the effect of each element of $\boldsymbol{\mu}$ on each element of $\boldsymbol{e}$ is calculated by taking partial derivatives at the current camera pose estimate:

$$J_{ij} = \frac{\partial e_i}{\partial \mu_j} \tag{1}$$

The motion vector may then be found as the solution of the equation

$$J\boldsymbol{\mu} = \boldsymbol{e} \tag{2}$$

The four steps of the control strategy used are further explained in the following sections.

3.2 Camera Pose and Motion Representation in SE(3)

A standard grey-scale CCD camera is used as a visual sensor. This is operated with a 4.2mm wide-angle lens at PAL field rate (50Hz, 768x288 image size.) The use of a wide-angle lens provides a large number of simultaneously trackable features, while the use of individual fields eliminates interlace tearing and increases the system's temporal resolution. A typical captured image is displayed as Step 1 of Figure 1.

Camera pose is represented by the matrix E which transforms points in world coordinates to camera coordinates:

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = E \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{3}$$

E takes the form

$$E = \begin{bmatrix} & R & & \boldsymbol{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where R is a rotation matrix ($|R| = 1$, $R^T R = I$) and $\boldsymbol{t}$ is a translation vector. The set of all possible values of E forms a representation of the 6-dimensional Lie Group SE(3), the group of rigid body transformations in $\mathbb{R}^3$. Once every video field, an update to camera pose is performed by means of left-multiplication by a motion matrix:

$$E_{t+1} = M_t E_t \tag{5}$$

where M takes the same form as E and represents the motion between fields. M may be parameterised by a six-dimensional motion vector via the exponential map [26]: for a given motion vector $\boldsymbol{\mu}$ the corresponding motion matrix is given by

$$M = \exp\left(\sum_{j=1}^{6} \mu_j G_j\right) \tag{6}$$

where G_j are the group generator matrices. Choosing μ_1, μ_2 and μ_3 to represent translation along the x, y and z axes and μ_4, μ_5 and μ_6 to describe rotation around these axes, the generator matrices take the values

$$G_1 = \begin{bmatrix} 0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix}, \; G_2 = \begin{bmatrix} 0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0 \end{bmatrix}, \; G_3 = \begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0 \end{bmatrix},$$
$$G_4 = \begin{bmatrix} 0&0&0&0\\0&0&1&0\\0&-1&0&0\\0&0&0&0 \end{bmatrix}, \; G_5 = \begin{bmatrix} 0&0&-1&0\\0&0&0&0\\1&0&0&0\\0&0&0&0 \end{bmatrix}, \; G_6 = \begin{bmatrix} 0&1&0&0\\-1&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix} \tag{7}$$

3.3 Projection

The wide-angle lens used exhibits a substantial amount of radial distortion, so a standard pin-hole camera model cannot be used directly. Radial distortion is approximated by a mapping of radii in normalised camera coordinates:

$$r' = r + \alpha r^3 + \beta r^5 \tag{8}$$

where $r = \sqrt{\left(\frac{x_c}{z_c}\right)^2 + \left(\frac{y_c}{z_c}\right)^2}$. Image pixel coordinates $\begin{pmatrix} u & v \end{pmatrix}^T$ are then given by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{bmatrix} \begin{pmatrix} \frac{r'}{r}\frac{x_c}{z_c} \\ \frac{r'}{r}\frac{y_c}{z_c} \\ 1 \end{pmatrix} \tag{9}$$

Where f_u, f_v represent focal length, u_0 the v_0 image center and α and β the cubic and quintic distortion terms of the camera. The procedure used to calibrate these parameters is described in Section 3.6.

3.4 Rendering and Image Measurements

Image errors are measured by comparing edges found in the video feed with edges rendered using the camera pose estimate. A CAD model containing the salient edges of the world (or object to be tracked) is required. This CAD model may also contain faces to allow the removal of hidden lines.

Hidden line removal is performed using z-buffered accelerated OpenGL. The first stage of rendering draws all model faces into the z-buffer. This is done without radial distortion, as this is not supported by the OpenGL pipeline.

Next, control points are initialised at regular intervals ($\sim$20 pixels) along every model edge. Each control point is tested for visibility against the z-buffer: if it is occluded, it is discarded. If it is visible, it is re-projected using the full radial distortion model and rendered to screen. An edge rendering made using visible distorted control points is shown as Step 2 of Figure 1.

At every visible control point, a one-dimensional edge detection in the direction of the model edge normal is performed in the video image. The closest intensity edge detected within a cutoff distance of the control point ($\sim$20 pixels) is assumed to be the video image edge corresponding to the current model edge. The pixel distance to this detected edge is inserted as a measurement into the error vector $\boldsymbol{e}$. Control points which do not detect an edge are discarded. These edge distance measurements are illustrated as Step 3 of Figure 1.

3.5 Robust Motion Vector Computation

Having filled the error measurement vector $\boldsymbol{e}$, the system next calculates the Jacobian matrix J. Each individual measurement e_i corresponds to a pixel distance from the ith successful control point to the nearest video edge along the model edge normal $\hat{\boldsymbol{n}}_i$. The Jacobian is therefore filled with partial differentials of the form

$$J_{ij} = \frac{\partial e_i}{\partial \mu_j} = \hat{\boldsymbol{n}}_i \cdot \begin{pmatrix} \frac{\partial u}{\partial \mu_j} \\ \frac{\partial v}{\partial \mu_j} \end{pmatrix} \tag{10}$$

With u and v being the projected image coordinates of the corresponding sample point. Their partial differentials with respect to the motion parameters are found by differentiating Equation (6) noting that

$$\frac{\partial M}{\partial \mu_j} = G_j \tag{11}$$

when evaluated at the origin ($\boldsymbol{\mu} = \mathbf{0}$). The chain rule is then used to differentiate Equations (5), (3), (8) and (9). Having found the error vector $\boldsymbol{e}$ and associated Jacobian J, Equation (2) may be solved for the motion parameters $\boldsymbol{\mu}$. The simplest least-squares solution could be found by the pseudo-inverse:

$$\boldsymbol{\mu} = J^{\dagger}\boldsymbol{e} = [J^T J]^{-1} J^T \boldsymbol{e} \tag{12}$$

however this solution is not very robust. Due to noise, misdetection of features and occlusion, the statistics of $\boldsymbol{e}$ are significantly non-Gaussian and thus a direct least-squares solution is inappropriate. Instead, a robust estimation for $\boldsymbol{\mu}$ is performed by finding the least square solution of

$$\begin{bmatrix} WJ \\ P \end{bmatrix} \boldsymbol{\mu} = \begin{pmatrix} W\boldsymbol{e} \\ \mathbf{0} \end{pmatrix} \tag{13}$$

where P and W are diagonal matrices of size 6×6 and $N \times N$ respectively, with N the size of the error vector. P is a stabilising regularisation matrix with elements inversely proportional to the prior standard deviation of each pose element. The matrix W is used to re-weight each measurement by a decaying function of e_j to obtain a robust M-estimator.

Having obtained $\boldsymbol{\mu}$, the tracking system then updates its pose estimate E with the corresponding motion matrix, and the servoing loop recommences. The updated pose is shown on the old video field as Step 4 of Figure 1.

3.6 Camera Calibration

Should the values of the intrinsic camera parameters used in Equations (8) and (9) not be known, these can be calibrated on-line. This is done by allowing the tracking system to modify intrinsic camera parameters just as it does the motion parameters. Representing the intrinsic camera parameters as a vector $\boldsymbol{p}$, with

$$\boldsymbol{p} = \begin{pmatrix} f_u & f_v & u_0 & v_0 & \alpha & \beta \end{pmatrix}^T \tag{14}$$

one can find the corresponding Jacobian J^p by differentiating Equations (8) and (9) with respect to the elements of $\boldsymbol{p}$:

$$J^p_{ij} = \frac{\partial e_i}{\partial p_j} = \hat{\boldsymbol{n}}_i \cdot \begin{pmatrix} \frac{\partial u}{\partial p_j} \\ \frac{\partial v}{\partial p_j} \end{pmatrix} \tag{15}$$

Adopting the superscript notation J^{μ} for the motion Jacobian, the tracking equation to be solved is then:

$$\begin{bmatrix} J^{\mu} & J^{p} \end{bmatrix} \begin{pmatrix} \boldsymbol{\mu} \\ \Delta\boldsymbol{p} \end{pmatrix} = \boldsymbol{e} \tag{16}$$

which may be solved robustly for $\begin{pmatrix} \boldsymbol{\mu}^T & \Delta\boldsymbol{p}^T \end{pmatrix}^T$ by re-weighted least squares just as in Section 3.5. Camera pose is updated as before, and camera parameters are simply updated as

$$\boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \Delta\boldsymbol{p} \tag{17}$$

For unknown camera parameters, a rough initial guess (optical center at image center, no distortion, f_v=$f_u/2$) of camera parameters is usually sufficient for the system to converge.

3.7 Pose Initialisation with Inertial Sensors

The maximum camera rotational velocity trackable by the visual sensor alone is between 0.3 and 1 rad/second, depending on the difficulty of the environment (clutter, edge contrast and density.) At high rotational velocities, image motion between frames may exceed the limited range edge detection; errors in edge correspondence may also occur. Finally, captured video images may be degraded by motion blur to the point where no edge detection is possible. To address these issues, three rate gyroscopes (Futaba G301) were mounted to the camera. These gyroscopes provide a voltage relative to rotational velocity. They are sampled at 171Hz using a micro-controller with a serial link. The sensor fusion strategy employed to combine visual and inertial measurements is illustrated in Figure 2 and is composed of three components: Motion prediction, bias correction and parametric edge detection.

Motion Prediction allows the tracking system to handle large un-blurred image motions. Since measuring the gyroscopes' output voltage over a serial link is fast compared to obtaining a video image from a frame-grabber, measurements from the inertial sensors can be used to predict camera pose for every new video field. This means that every field, regardless of the magnitude rotational velocity, the visual tracking system starts in almost the right place and has to perform only a small pose corrections. The problem of image features moving beyond the tracking system's search range due to rotation is hence eliminated.

A pose prediction is obtained by integrating the rotational velocities measured from the inertial sensors over the time from the previous to the current video field, and adjusting the camera pose by the resulting rotation. Translation

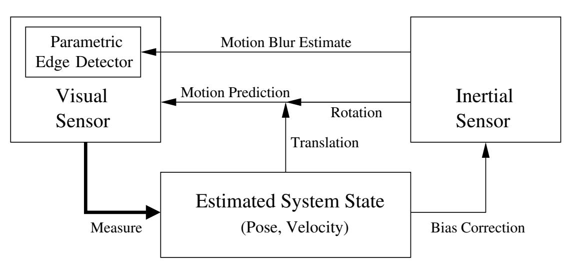

Figure 2. Sensor fusion strategy

is not measured from the sensors (no linear accelerometers are currently used) but is estimated from a simple motion model. The use of motion prediction alone greatly increases the tracking system's robustness, and rotational velocities of 0.8-3.6 rad/s are trackable, again depending on the difficulty of the tracked scene.

The voltage measured from the inertial sensors is a linear function function of angular velocity; the voltage produced at rest is known as the *bias voltage*, and must be subtracted from measurements to obtain rotational velocity. Unfortunately the bias can drift substantially with time and temperature variation. Besides calibrating the bias at the start of each tracking session, it is therefore necessary to continually correct the system's bias estimate. This is done by comparing rotation predictions made by the inertial sensors to the actual rotation measured by the visual sensor; if a discrepancy is found, the bias estimate is adjusted accordingly.

3.8 Parametric Edge Detection

Unless a camera's exposure time is infinitely short, images taken of while the camera is rotating may be affected by *motion blur*. While some cameras allow exposure time to be reduced to very small values, this can be at the expense of low light performance, depth-of-field and image noise. Furthermore, compact and lightweight cameras such as used as a head-mounted sensor may well not have a facility for adjusting exposure time, which may be as large as the duration of a full video field (20ms).

At high rotational velocities, image degradation due to motion blur is sufficient to make features undetectable by standard edge detection. Figure 3 shows a video field captured by a rotating camera. The camera rotation of 2.3 rad/s causes vertical edges to be spread out over 25 pixels. This can be modeled as a convolution which transforms image intensity steps into intensity ramps of constant gradient.

To maintain tracking in these conditions, the behaviour of the tracking system's edge detector is modified. The inertial sensors can provide an measurement of rotational velocity at the time of exposure. This is multiplied by an off-line measurement of camera exposure time to obtain an estimate of the rotation the camera undergoes during exposure. For

Figure 3. Motion blur due to camera rotation

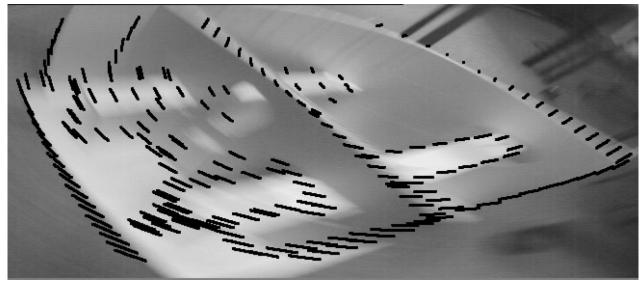

Figure 4. Predicted motion blur vectors

each control point, this rotation estimate can be multiplied by the differentials calculated for Equation (10) to obtain the image motion of the control point during exposure and the component of this motion in the edge normal direction. Figure 4 shows a blurred scene where the estimated motion blur vectors have been drawn at each control point. Once blur length in the edge normal direction has been predicted, a *matched filter* can be used to detect blurred edges. This filter is a zero-mean ramp of the same length as the predicted blur. The filter is convolved with the pixels along the edge normal, and the first large local maximum is the detected edge.

4 AR Display and Calibration

A Sony Glasstron HMD modified for frame-sequential stereo operation is used as an optical see-through AR display. This display has a resolution of 800×600 pixels and a 30° field-of-view. Figure 5 shows the display mounted on a helmet along with the sensor camera.

This section describes the projection model and rendering process used to draw geometry on the AR display. Further, this section describes how the projection parameters for a user are calibrated. It should be noted that each eye must be calibrated separately: there is no information shared between the left and right eyes. Hence the rest of this section (and indeed the coordinate frame transformation illustrated in Figure 5) will describe the monocular case only.

4.1 Projection Model

Projection of geometry to the display is performed by treating the display's LCD screen as the image plane of a virtual camera located at the user's eye. This allows normal computer graphics projection models to be used. However, we have found that the display exhibits a substantial amount of radial distortion (in this case, *pincushion* distortion) which cannot be modeled by a pin-hole camera alone, so we include a polynomial model of radial distortion.

The projection model used is very similar to that given in Section 3.3, however there are slight changes in terms of

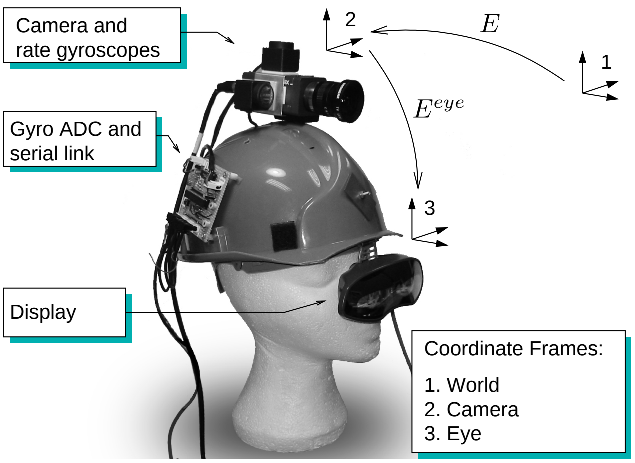

Figure 5. The AR display and associated coordinate systems

coordinate systems and the radial distortion approximation. The visual tracking system in Section 3 transforms coordinates from the world coordinate frame to the sensor camera frame, so the AR projection never encounters world coordinates. Instead, geometry is transformed from sensor camera frame to eye coordinate frame:

$$\begin{pmatrix} x_e \\ y_e \\ z_e \\ 1 \end{pmatrix} = E^{eye} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} \tag{18}$$

where E^{eye} takes the same form as E in Equation (4). Projection to the AR screen pixel coordinates $(\ u \quad v\ )^T$ is performed using the modified pin-hole camera model:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{bmatrix} \begin{pmatrix} \frac{r'}{r}\frac{x_e}{z_e} \\ \frac{r'}{r}\frac{y_e}{z_e} \\ 1 \end{pmatrix} \tag{19}$$

The radial distortion model used is:

$$r' = r + \alpha r^3 \tag{20}$$

where $r = \sqrt{\left(\frac{x_e}{z_e}\right)^2 + \left(\frac{y_e}{z_e}\right)^2}$. The quintic term βr^5 found in Equation (8) is not included to aid convergence with a small number of measurements. As the radial distortion produced by the AR display is not as severe as that produced by the tracking lens, this simpler approximation suffices.

4.2 Rendering

Rendering is performed using accelerated z-buffered OpenGL. The display of lines and textured triangles is supported. For the evaluation of the AR display, the same CAD model as used for tracking was drawn, using black triangles (which appear transparent in the display) for hidden line removal. Rendering the edges of the tracking CAD model thus allows immediate visual inspection of the accuracy of tracking and calibration.

Radial distortion is not directly supported in the OpenGL rendering transformation, and therefore requires a separate step. In Section 3.4 radial distortion was performed on a point-by-point basis: only a few hundred points needed to be distorted very accurately, with no need for smoothly distorted lines or triangles. The requirements for AR rendering are different; the rendering of arbitrarily complex meshes may be required. Further, long line segments and large textured triangles should also be displayed with correct distortion, so a by-vertex distortion scheme is inappropriate.

Instead, the approach presented by Watson and Hodges [27] is used. An undistorted projection of the scene is first rendered into an off-screen buffer using full hardware acceleration. Radial distortion can then be applied by rendering a distorted grid to the screen using this buffer as a texture map. The vertices of the grid are pre-distorted with the inverse of the HMD's optical distortion to produce an undistorted view for the user. A 10x10 grid has proven sufficiently accurate to avoid discontinuities at the grid cell boundaries.

4.3 User Calibration Procedure

When a user dons the AR headset and starts the tracking system, the augmented features rendered are likely to be significantly misaligned with their intended real-world locations. This means the projection parameters used in Section 4.1 are not correct for the user. This is in part due to the different geometry of each user's head, as every user will have a different offset between sensor camera and eyes. However, even for a single user, a system perfectly aligned during one session can be very much misaligned when starting the next, as the AR display will not be in exactly the same place relative to the user's eyes. Very slight movements of the display relative to the eyes can have a large impact on the display's optical center and indeed on the magnitude of radial distortion encountered; focal length and modeled eye position are also affected.

Therefore, it is currently necessary to *calibrate* the display for every single use. Calibration is the procedure by which appropriate values for the projection parameters used in Section 4.1 are determined. In video see-through augmented reality calibration can be performed automatically by comparing the AR projection to the video signal. In contrast, the calibration of an optical see-through display must rely on the user for input.

The user is provided with a mouse which controls a cross-hairs visible in the AR display. To calibrate the

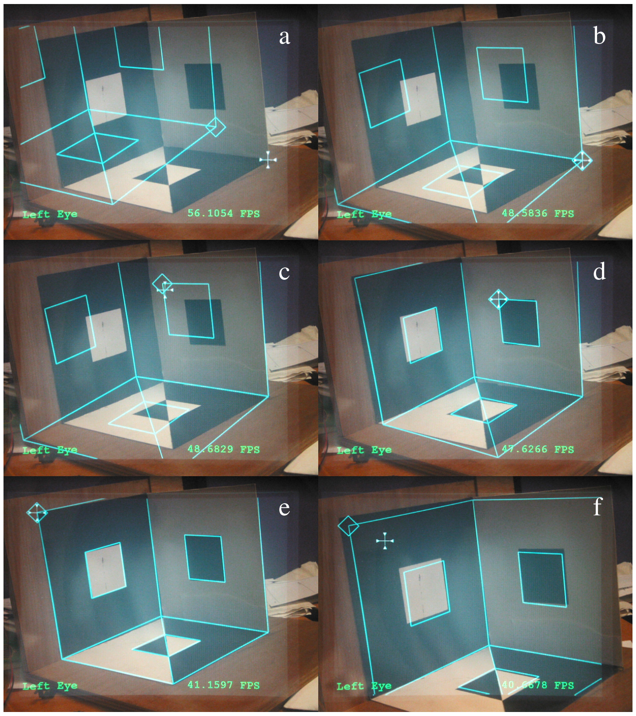

Figure 6. Calibration seen through the display

projection parameters, the user is asked to align projected model vertices with their real-world location. To do this, the user first selects a projected vertex with the left mouse button, and then clicks on its real-world position with the right mouse button. To reduce the negative impact of tracking jitter and user head motion on measurement accuracy, information from the tracking system is used to motion-stabilise the cross-hairs with respect to the projected geometry.

To ensure a good calibration, the user is encouraged to make measurements at many different places in the display, including the center and the extremities. Also, the user is encouraged to make measurements using vertices at many different distances - this is crucial for proper stereo alignment. If the tracking environment contains features at many depths, this is simply done by selecting some vertices which are near and some which are far away; if the environment is shallow, the user is encouraged to move nearer and further from the model during the course of the calibration. Generally, the user will make five measurements at one range, then five at a different depth, and then make further measurements to clean up any remaining misaligned vertices as necessary. Figure 6 illustrates this procedure in six panels. The view in (a) is completely uncalibrated. The user has selected a vertex to adjust (indicated by a diamond) and moves the cross-hairs to the real-world position. The user clicks the right mouse button and the calibration is updated accordingly (b). The user then selects the next vertex (c) and clicks on the corresponding position (d). After three more measurements (for a total of five) the view is well aligned from this viewpoint (e); however, moving the closer to the target reveals errors (f) and more measurements will have to be made at this range. It should be pointed out that the very limited range of movement shown is due to constraints imposed by photography through the AR display, and a real user's head would move about more.

4.4 Nonlinear Optimisation

Each user measurement provides the system with two pieces of information; a 2D display coordinate $\begin{pmatrix} \tilde{u} & \tilde{v} \end{pmatrix}^T$ specified by the user's cross-hairs, and a 3D sensor-camera coordinate $\begin{pmatrix} x_c & y_c & z_c & 1 \end{pmatrix}^T$ which was the selected vertex's camera-frame position at the time of the measurement. The ideal HMD calibration would project all the camera-frame coordinates to their measured display positions. Due to measurement noise and model inaccuracies, this is not achievable, and projection parameters are chosen to minimise the sum-squared re-projection error

$$\epsilon = \sum_{n=1}^{n=N} (\tilde{u}_n - u_n)^2 + (\tilde{v}_n - v_n)^2 \tag{21}$$

where N is the number of measurements made, and $\begin{pmatrix} u_n & v_n \end{pmatrix}^T$ is the projection of the nth sensor-camera coordinate. Re-writing the above in terms of an error vector $\boldsymbol{e}$,

$$\epsilon = |\boldsymbol{e}|^2, \qquad \boldsymbol{e} = \begin{pmatrix} \tilde{u}_1 - u_1 \\ \tilde{v}_1 - v_1 \\ \vdots \\ \tilde{u}_N - u_N \\ \tilde{v}_N - v_N \end{pmatrix} \tag{22}$$

This minimisation of this error is equivalent to the tracking problem described in Section 3.6. Writing the intrinsic projection parameters as a vector $\boldsymbol{p}$ with

$$\boldsymbol{p} = \begin{pmatrix} f_u & f_v & u_0 & v_0 & \alpha \end{pmatrix}^T \tag{23}$$

and using a vector $\boldsymbol{\mu}$ to update the camera-to-eye transformation with the equation,

$$E_{t+1}^{eye} = \exp\left(\sum_{j=1}^{6} \mu_j G_j\right) E_t^{eye} \tag{24}$$

the reprojection error is minimised by solving

$$\begin{bmatrix} J^{\mu} & J^{p} \end{bmatrix} \begin{pmatrix} \boldsymbol{\mu} \\ \Delta\boldsymbol{p} \end{pmatrix} = \boldsymbol{e} \tag{25}$$

where

$$J_{ij}^{\mu} = \frac{\partial e_i}{\partial \mu_j} \qquad J_{ij}^{p} = \frac{\partial e_i}{\partial p_j} \tag{26}$$

with differentials calculated using the chain rule as before.

In contrast to Section 3.5 where measurements were significantly non-Gaussian, the user's measurements can be considered less prone to clutter and feature mis-detection and so the error terms here are considered to follow a Gaussian distribution. Hence the pseudo-inverse solution to the above equation is used:

$$\begin{pmatrix} \boldsymbol{\mu} \\ \Delta \boldsymbol{p} \end{pmatrix} = \lambda \begin{bmatrix} J^{\mu} & J^{p} \end{bmatrix}^{\dagger} \boldsymbol{e} \qquad (27)$$

The scale factor $\lambda \approx 0.1$ slows down convergence, aiding numerical stability and allowing the user to observe convergence in the display.

5 Results

5.1 Tracking System Performance

Using inertial sensors for pose initialisation and motion blur compensation, the visual tracking system is capable of tracking camera rotations of up to 3.1-4.7 rad/s, depending on scene difficulty. This corresponds to moderately rapid head motion. The scenes tracked contained features blurred over a length of 33-50 pixels.

5.2 Calibration Performance

The static performance of the HMD calibration method used was evaluated by performing eight full stereo calibrations on the test model shown in Figure 6. On average, twelve measurements were made for each eye. The mean residual error for all measurements is shown in Table 1. To evaluate the effect of including radial distortion in the projection model, each calibration was also re-calculated without radial distortion. This resulted in a residual error which was on average greater by 0.7 pixels.

Subjectively, registration errors with distortion enabled appeared smaller than 1cm when viewing the object from a distance of 50-150 cm. With distortion disabled, monocular registration errors were similar, however the effect on stereo perception was significant; graphics often appeared to float a few centimeters in front of the real object. This is supported by the numerical calibration results. On average, removing radial distortion from the projection model caused calibrated left-right eye separation to increase by 3mm, and also caused calibrated eye centers to move 1.8cm further forwards.

5.3 Dynamic Registration Error

Since both visual tracking and augmented rendering currently require accelerated OpenGL, the system is currently distributed across two networked computers. One machine performs tracking and a second machine renders the augmented visuals and provides the UI. The latency caused by this split (which will be addressed in future) dominates the system's dynamic registration error, and rendered visuals lag actual motion by $\sim$140ms. Nonetheless, the effect of pre-distorting the augmented rendering may be considered. Holloway [12] argues that the static registration improvements gained by modeling radial distortion can be outweighed by the extra latency incurred. Table 1 shows how rendering performance changes when distortion is enabled. Even though rendering is always close to the display's maximum frame-rate of 30 frames/second/eye, the distortion step contributes 13-17 ms of latency to the system. It should be noted however that modern graphics accelerators offer substantially higher fill-rates and texture bandwidth than the card used here (NVidia GeForce 2MX).

Distortion Enabled	Yes	No
Mean Reprojection Error (pixels)	3.4	4.1
Rendering Speed (Frames/sec/eye)		
Simple scene:	40	83
Textured scene:	27	49

Table 1. Effect of radial distortion on calibration and rendering performance

6 Conclusions and Limitations

This paper has presented a flexible and robust hybrid tracking system for augmented reality. The system is flexible in that it does not depend on external sensors or fiducial markers and can track motion of a workpiece as well as motion of a user. The system is robust enough to track reasonably rapid user head rotation, even in the face of substantial image degradation due to motion blur.

Further, this paper has demonstrated an AR calibration and rendering system capable of calibrating and correcting distortion found in optically see-through displays. While the magnitude of distortion exhibited by the display used here was not large, the same method could be applied to AR displays with larger fields-of-view (and hence larger amounts of distortion.)

While the tracking system presented is capable of operating in completely un-instrumented environments (the instrumentation is entirely worn by the user), the lack of uniquely distinguishable and easily locatable beacons or templates does impose limitations on the system's operation. Notably, automatic localisation is not supported, and the system must be initialised from an approximately correct (within 20 pixels) starting point. This initialisation must also be performed whenever tracking is lost. Further, approximate symmetries or repetitions in the model (e.g., the squares of a chess-board) can cause tracking to converge on incorrect local minima, especially when the image contains many parallel lines in close proximity. The system is further limited to operation in such environments which can accurately be represented as a CAD model of edges; many natural features such as bushes and trees or environments which are constantly changing are therefore not supported.

References

[1] T. Auer, S. Brantner, and A. Pinz. The integration of optical and magnetic tracking for multi-user augmented reality. In M. Gervaut, D. Schmalstieg, and A. Hildebrand, editors, *Virtual Environments '99. Proceedings of the Eurographics Workshop in Vienna, Austria*, pages 43–52, 1999.

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. Macintyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, pages 34–47, November 2001.

[3] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through hmd. In *Proc. SIGGRAPH '94*, pages 197–204, 1994.

[4] R. Behringer, J. Park, and V. Sundareswaran. Model-based visual tracking for outdoor augmented reality. In *Proc. IEEE and ACM ISMAR'02*, Darmstadt, Germany, September 2002.

[5] L. Chai, K. Nguyen, W. Hoff, and T. Vincent. An adaptive estimator for registration in augmented reality. In *Proc. IEEE and ACM IWAR'99*, pages 20–21, October 1999.

[6] T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. In *Proc. British Machine Vision Conference (BMVC'99)*, volume 2, pages 574–583, Nottingham, 13–16 September 1999.

[7] B. Espiau and F. Chaumette. A new approach to visual servoing. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.

[8] Y. Genc, S. Riedel, F. Souvannavong, C. Akinlar, and N. Navab. Marker-less tracking for AR: A learning-based approach. In *Proc. IEEE and ACM ISMAR'02*, Darmstadt, Germany, September 2002.

[9] Y. Genc, F. Sauer, F. Wenzel, M. Tuceryan, and N. Navab. Optical see-through hmd calibration: A novel stereo method validated with a video see-through system. In *Proc. IEEE and ACM ISAR'00*, pages 165–174, Munich, October 2000.

[10] C. Harris. Tracking with rigid models. In A. Blake, editor, *Active Vision*, chapter 4, pages 59–73. MIT Press, 1992.

[11] W. Hoff, K. Nguyen, and T. Lyon. Computer vision-based registration techniques for augmented reality. *Intelligent Robots and Computer Vision XV*, 2904:538–548, 1996.

[12] R. Holloway. *Registration Errors in Augmented Reality Systems*. PhD thesis, University of North Carolina at Chapel Hill, 1995.

[13] M. Kanbara, H. Fujii, H. Takemura, and N. Yokoya. A stereo vision-based mixed reality system with natural feature point tracking. In *The Second International Symposium on Mixed Reality (ISMR'01)*, pages 56–63, March 2001.

[14] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. In *Proc. British Machine Vision Conference (BMVC'02)*, volume 2, pages 787 –796. BMVA, September 2002.

[15] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time Vision-Based camera tracking for augmented reality applications. In D. Thalmann, editor, *Proc. ACM Symposium on Virtual Reality Software and Technology*, New York, NY, 1997. ACM Press.

[16] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, August 1992.

[17] E. Marchand and F. Chaumette. A new formulation for non-linear camera calibration using virtual visual servoing. Technical Report 1366, IRISA, 2001.

[18] E. Marchand and F. Chaumette. Virtual visual servoing: a framework for real-time augmented reality. In G. Drettakis and H. Seidel, editors, *Proc. Eurographics 2002*, volume 21, pages 289–298, Saarbrcken, Germany, September 2002.

[19] J. Park, S. You, and U. Neumann. Natural feature tracking for extendible robust augmented realities. In *Proc. IEEE IWAR'98*, San Francisco, November 1998.

[20] H. Rehbinder and B. Ghosh. Multi-rate fusion of visual and inertial data. In *Proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 97–102, 2001.

[21] J. Rolland, L. Davis, and Y. Baillot. A survey of tracking technologies for virtual environments. In W. Barfield and T. Caudell, editors, *Fundamentals of Wearable Computers and Augmented Reality*. Lawrence Erlbaum Assoc, 2000.

[22] A. State, G. Hirota, D. Chen, W. Garrett, and M. Livingston. Superior augmented reality tracking by integrating landmark tracking and magnetic tracking. In *Proc. SIGGRAPH'96*, pages 429–438, 1996.

[23] V. Sundareswaran and R. Behringer. Visual servoing-based augmented reality. In *Proc. IEEE IWAR'98*, San Francisco, November 1998.

[24] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[25] M. Tuceryan and N. Navab. Single point active alignment method (SPAAM) for optical see-through HMD calibration for ar. In *Proc. IEEE and ACM International Symposium on Augmented Reality (ISAR'00)*, pages 149–158, Munich, October 2000.

[26] V. Varadarajan. *Lie Groups, Lie Algebras and Their Representations*. Number 102 in Graduate Texts in Mathematics. Springer-Verlag, 1974.

[27] B. Watson and F. Hodges. Using texture maps to correct for optical distortion in head-mounted displays. In *Proc. IEEE Virtual Reality Annual Symposium (VRAIS'95)*, pages 172–178, 1995.

[28] Y. Yokokohji, Y. Sugawara, and T. Yoshikawa. Accurate image overlay on see-through head-mounted displays using vision and accelerometers. In *Proc. IEEE Conference on Virtual Reality*, pages 247–254, 2000.

[29] S. You and U. Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *Proc. IEEE Conference on Virtual Reality*, pages 71–78, March 2001.