Sensor Fusion and Occlusion Refinement for Tablet-based AR

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

Abstract

This paper presents a set of technologies which enable robust, accurate, high resolution augmentation of live video, delivered via a tablet PC to which a video camera has been attached. By combining several technologies this is achieved without the use of contrived markers in the environment: An outside-in tracker observes the tablet to generate robust, low-accuracy pose estimates. An inside-out tracker running on the tablet observes the video feed from the tablet-mounted camera and provides high accuracy pose estimates by tracking natural features in the environment. Information from both of these trackers is combined in an Extended Kalman Filter. Finally, to maximise the quality of the augmented imagery, boundaries where the real world occludes the virtual imagery are identified and another tracker is used to refine the boundaries between real and virtual imagery so that their synthesis is as convincing as possible.

1. Introduction

Augmented Reality (AR) is the synthesis of real and virtual imagery. In contrast to Virtual Reality where the user is fully immersed in virtual imagery, AR applications fundamentally require the user to be aware of and often interact with their physical surroundings. In an ideal world, an AR system should be able to augment a user's direct view of the world with high resolution graphics over a wide field of view with no error, latency or jitter. Currently this technology is not available (or if it is, a suitable academic discount is not offered) and hence a variety of technologies are used each of which makes a different compromise. Optical see-through head-mounted displays offer a high resolution view of the real world, but usually suffer from latency or jitter in the virtual imagery and require calibration for each user. Video feed-through HMDs remove the need for per-user calibration, but are forced to render the world at low resolution with latency. Both classes greatly restrict the user's field-of-view.

Small, portable displays such as PDAs offer an alternative to this approach. Rather than augmenting the user's view of the world directly, they act as the viewfinder for a video camera and operate by augmenting the video feed as it is displayed (in a similar manner to video feed-through systems). The great advantage is that here, small latencies do not matter and further, a hand-held device can be less obtrusive than a head-mounted one. Unfortunately the processing power and bandwidth available on PDAs is limited while tablet PCs offer the performance required for systems like ours today. Further, the large screen available on a tablet provides a better medium for high resolution augmentations.

This paper demonstrates that by combining several tracking technologies, it is possible to deliver robust and accurate live video augmentation via a tablet PC without the use of artificial fiducials in the scene. After reviewing related work in the field, this paper first introduces a mathematical framework which allows a transparent and flexible manipulation of pose, motion and uncertainty information. This framework is described in Section 3.

Next, an edge based tracking system capable of real-time operation on a tablet PC is presented (Section 4). This system employs a CAD model of salient edges which allows it to produce highly accurate pose measurements for subsequent video-see through augmentation. While this system is very accurate for estimating small pose changes, it cannot cope with sudden large camera motions. Therefore, a robust outside-in LED tracker is employed to provide absolute measurements (Section 5.) This tracker employs a novel correspondence algorithm to identify six LEDs mounted on the back of the tablet. To correctly merge the two sources of pose information, Section 6 describes the implementation of an Extended Kalman Filter in terms of the mathematical framework described. To evaluate the performance of the tracking strategy described, a simple AR entertainment application is described in Section 7.

Section 8 describes an algorithm to improve the quality of the augmented visuals by increasing the accuracy with which virtual objects are occluded by real world geometry. 3D model information available to the edge-based tracker is exploited to refine individual occluding edges in the captured video image. Finally, section 9 presents results and conclusions.

2. Background

Few tablet-based systems have been presented in the literature. Vlahakis *et al.* use a tablet as part of the larger ARCHEOGUIDE project [16]. ARCHEOGUIDE aims to enhance the experience of visiting archaeological sites e.g. by rendering reconstructions of historic places in their original locations. The project investigates multiple types of AR apparatus: Head-mounted displays are used for video see-through augmentations, while tablet and pocket PCs are used as replacements for paper guides. The tablets are equipped with differential GPS and a compass, and can replay multimedia streams appropriate to the user's location.

Zhu, Owen *et al.* present the PromoPad, an augmented reality shopping assistant [23] using video see-through augmentations. Registration is marker-based and based on the Owen's earlier work on ARToolkit markers [12]. Emphasis is placed on the PromoPad's ability to detect context from the user's location, and to update the presented information accordingly. For example, products known to be of interest to a customer can be highlighted, while less desirable products can be hidden through (diminished reality).

Recent research into PDA-based AR has often been focused on enabling technologies. Notably, a port of the popular AR Toolkit to the pocket PC platform has been presented by Wagner *et al.* [18]. Although pure pose estimation performance of up to 15 frames/sec is possible on an XScale CPU, current cameras cannot supply more than 8 frames/sec to the CPU; nevertheless, fully self-tracking PDA applications using this framework have been demonstrated [17]. MacWilliams *et al.* demonstrate that PDAs can in theory coexist with conventional workstation-based AR in a distributed AR framework [11], but have encountered some rendering performance issues.

The combination of inside-out and outside-in tracking has recently been studied by Satoh *et al.* in [14]. Inside-out fiducial tracking is combined with outside-in tracking of single point-like head markers . For the case of only one head marker, measurements are combined by constraining the pose output from the inside-out tracking to lie on the line from the outside camera through the head marker. If more than one head marker is used the sum squared re-projection error of all markers in all cameras is used. The transformation from the head markers to head-mounted camera, as well as the position of the outside-in camera, are assumed to be known. Baillot *et al.* demonstrate a method for calibrating these transformations on-line from a few motion correspondences in both reference frames [1].

Early approaches to determining the occlusion of real and virtual objects include those of Wloka and Anderson [22] and Breen *et al.* [4]. Both use stereo cameras to estimate a depth map of the user's view, and use this depth map to handle the occlusion of augmented visuals. Wloka motivates this approach by pointing out that in most AR applications, the assumption of an unchanging scene is unrealistic because of e.g. the user's interaction. Breen instead studies the static case further by replacing the on-line depth map with pre-registered 3D models of occluding real objects.

Berger [3] does not explicitly estimate depth and does not use 3D models of real occluding geometry. Rather, an estimate of occlusion regions is built by tracking 2D contours found in the image. By observing the motion of these countours over time, they can be labelled as being "in front of" or "behind" virtual imagery. Impressive results are presented for scenes without complex textures. Lepetit and Berger [10] later extend this idea to a high-accuracy off-line scenario; by tracking using user-seeded occluding curves through a video sequence, a 3D reconstruction of the occluding object is computed. The resulting accurate segmentation of the sequence into foreground and background allows virtual objects to be inserted into the video with high precision.

Recent real-time work occlusion has focused on real objects dynamically occluding virtual ones. Fuhrmann *et al.* use maker-based human motion capture to register occluding users in a collaborative scene [6]. A transparent 3D humanoid model aligned to the motion-capture pose is rendered into the z-buffer to occlude scientific visualisations in a collaborative application. This "phantom" is blurred for regions which are not directly tracked, e.g. the fingers of user's hands. Fischer *et al.* detect objects occluding objects by learning the textures of a static surrounding scene. By comparing video input to a textured predicted re-projection, occluders can be identified. Further work on the application of depth-from-stereo to AR occlusion has been presented by Kanbara *et al.* [7].

3. Mathematical Framework

This section briefly introduces the mathematical framework employed. Points in 3D space are represented as homogeneous coordinates of the form $(x\ y\ z\ 1)^T$. Points are transformed from coordinate frame $\mathcal{A}$ to frame $\mathcal{B}$ by left-multiplication with a 4×4 Euclidean transformation matrix denoted E:

$$\begin{pmatrix} x_{\mathcal{B}} \\ y_{\mathcal{B}} \\ z_{\mathcal{B}} \\ 1 \end{pmatrix} = E_{\mathcal{B}\mathcal{A}} \begin{pmatrix} x_{\mathcal{A}} \\ y_{\mathcal{A}} \\ z_{\mathcal{A}} \\ 1 \end{pmatrix} \tag{1}$$

where the subscript $\mathcal{B}\mathcal{A}$ may be read as "$\mathcal{B}$ from $\mathcal{A}$". The product $E_{\mathcal{C}\mathcal{A}} = E_{\mathcal{C}\mathcal{B}}E_{\mathcal{B}\mathcal{A}}$ transforms points from frame $\mathcal{A}$ to $\mathcal{C}$ and the transformations have an inverse $E^{-1}_{\mathcal{A}\mathcal{B}} = E_{\mathcal{B}\mathcal{A}}$. The matrices take the form

$$E = \left[\begin{array}{cc} R & \boldsymbol{t} \\ 0\ 0\ 0 & 1 \end{array} \right] \tag{2}$$

where R is a rotation matrix ($|R| = 1$, $R^T R = I$) and $\boldsymbol{t}$ is a translation vector. The set of all possible E forms a representation of the 6-dimensional Lie Group SE(3), the group of rigid body transformations in $\mathbb{R}^3$.

With time, the transformations between coordinate frames may change and such a change is represented with a motion matrix denoted M:

$$E_{\mathcal{B}\mathcal{A}|t+1} = M_{\mathcal{B}} E_{\mathcal{B}\mathcal{A}|t} \tag{3}$$

where $M_{\mathcal{B}}$ represents motion in frame $\mathcal{B}$ and takes the same form as E in Eq. (2). M is parametrised by a six-dimensional motion vector $\boldsymbol{\mu}$ via the exponential map: for a given motion vector $\boldsymbol{\mu}_{\mathcal{B}}$ in frame $\mathcal{B}$ the corresponding motion matrix is given by

$$M_{\mathcal{B}} = \exp(\boldsymbol{\mu}_{\mathcal{B}}) \equiv e^{\sum_{j=1}^{6} \mu_{\mathcal{B}j} G_j} \tag{4}$$

where G_j are the group generator matrices[1]. 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

$$G_1 = \begin{bmatrix} 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_2 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_3 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{bmatrix},$$
$$G_4 = \begin{bmatrix} 0&0&0&0 \\ 0&0&1&0 \\ 0&-1&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_5 = \begin{bmatrix} 0&0&-1&0 \\ 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_6 = \begin{bmatrix} 0&1&0&0 \\ -1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix} \tag{5}$$

Using this formulation, complex coordinate frame transformations are easily differentiable:

$$\frac{\partial}{\partial \mu_{\mathcal{B}j}} \left(E_{\mathcal{C}\mathcal{B}} M_{\mathcal{B}} E_{\mathcal{B}\mathcal{A}} \right) = E_{\mathcal{C}\mathcal{B}} G_j E_{\mathcal{B}\mathcal{A}}. \tag{6}$$

Motions can be transformed from one coordinate frame to another either as matrices

$$M_{\mathcal{A}} = E_{\mathcal{A}\mathcal{B}} M_{\mathcal{B}} E_{\mathcal{B}\mathcal{A}} \tag{7}$$

[1] Closed forms of both the exponential and the log exist. Further information on the Lie Group SE(3) and its properties may be found in [15].

or as motion vectors using the *adjoint operator*. The adjoint of a transformation matrix yields a 6×6 matrix such that

$$\boldsymbol{\mu}_{\mathcal{A}} = \mathrm{Adj}(\mathrm{E}_{\mathcal{A}\mathcal{B}}) \boldsymbol{\mu}_{\mathcal{B}} \tag{8}$$

and takes the value (writing the cross operator $\wedge$)

$$\mathrm{Adj}(\mathrm{E}) = \begin{bmatrix} R & \boldsymbol{t} \wedge R \\ 0 & R \end{bmatrix}. \tag{9}$$

Often, the true value of a transformation matrix is unknown and only a noisy estimate $\hat{E}$ can be obtained. In the same way as motions are defined in a specific coordinate frame, errors are also relative to a frame. Choosing w.l.o.g. to represent errors in frame $\mathcal{B}$, the relationship between estimate and true state is written

$$\hat{E}_{\mathcal{B}\mathcal{A}} = \exp(\boldsymbol{\epsilon}_{\mathcal{B}}) E_{\mathcal{B}\mathcal{A}} \tag{10}$$

where the error 6-vector $\boldsymbol{\epsilon}$ is normally distributed:

$$\boldsymbol{\epsilon}_{\mathcal{B}} \sim N(\mathbf{0}, \Sigma_{\mathcal{B}}). \tag{11}$$

Here $\Sigma_{\mathcal{B}}$ is the estimate's 6×6 covariance matrix in coordinate frame $\mathcal{B}$. It is sometimes desirable to transform covariance matrices from one coordinate frame to another: writing e.g. $\hat{E}_{\mathcal{C}\mathcal{A}} = E_{\mathcal{C}B}\hat{E}_{\mathcal{B}\mathcal{A}}$, errors transform like motions in Eq. (8)

$$\boldsymbol{\epsilon}_{\mathcal{C}} = \mathrm{Adj}(\mathrm{E}_{\mathcal{C}\mathcal{B}}) \boldsymbol{\epsilon}_{\mathcal{B}} \tag{12}$$

for any sample from the error distribution. It can be shown that covariance matrices also transform with the adjoint:

$$\Sigma_{\mathcal{C}} = \mathrm{Adj}(\mathrm{E}_{\mathcal{C}\mathcal{B}}) \Sigma_{\mathcal{B}} \mathrm{Adj}(\mathrm{E}_{\mathcal{C}\mathcal{B}})^{\mathrm{T}}. \tag{13}$$

This is shown by equating probabilities for the mapped distribution,

$$\frac{1}{|\mathrm{Adj}(E_{\mathcal{B}})|} \frac{e^{-\frac{1}{2}\boldsymbol{\epsilon}_{\mathcal{C}}^T \Sigma_{\mathcal{C}}^{-1} \boldsymbol{\epsilon}_{\mathcal{C}}}}{\sqrt{(2\pi)^6 |\Sigma_{\mathcal{C}}|}} = \frac{e^{-\frac{1}{2}\boldsymbol{\epsilon}_{\mathcal{B}}^T \Sigma_{\mathcal{B}}^{-1} \boldsymbol{\epsilon}_{\mathcal{B}}}}{\sqrt{(2\pi)^6 |\Sigma_{\mathcal{B}}|}} \tag{14}$$

and noting that $\forall E$, $|\mathrm{Adj}(E)| = 1$ and $(\mathrm{Adj}(E)\mathbf{0} = \mathbf{0}) \implies (|\Sigma_{\mathcal{B}}| = |\Sigma_{\mathcal{C}}|)$ so all denominators cancel. The result in Eq. (13) is obtained by substituting Eq. (12) and rearranging.

If the errors in 10 were represented in frame $\mathcal{A}$ instead of frame $\mathcal{B}$,

$$\hat{E}_{\mathcal{B}\mathcal{A}} = E_{\mathcal{B}\mathcal{A}} \exp(\boldsymbol{\epsilon}_{\mathcal{A}}) \tag{15}$$

then the distribution of $\boldsymbol{\epsilon}_{\mathcal{A}}$ is different from that of $\boldsymbol{\epsilon}_{\mathcal{B}}$. For example, an ambiguity in rotation in coordinate frame $\mathcal{B}$ corresponds to a coupled translation and rotation ambiguity in coordinate frame $\mathcal{A}$. For this reason it is necessary to know how to transform covariance matrices from one coordinate frame to another.

In the remainder of this paper, the following coordinate frames are used:

$\mathcal{W}$	:	World (i.e. Model)
$\mathcal{C}$	:	Tablet-mounted camera, (z=optical axis)
$\mathcal{T}$	:	Back of tablet (x, y in plane of tablet)
$\mathcal{S}$	:	Camera (sensor) observing tablet

4. Inside-out edge tracking

This section describes an edge-based tracking system by which the pose of a tablet-mounted camera is tracked. The edge-based tracking system employed has previously been applied to HMD-based AR [9]; what follows here is a brief review of the system's operation. Video at 640×480×8bpp and 30Hz is captured by a fire-wire camera mounted to the top of the tablet.

The tracking system employed relies on the availability of a 3D model of the scene to be tracked. This 3D model must describe salient edges and any occluding faces. Using a predicted estimate of camera pose, an estimate of the tablet camera's view of the model can be projected at every frame: a point in the world reference frame $\boldsymbol{x}_{\mathcal{W}} = (x_{\mathcal{W}}\ y_{\mathcal{W}}\ z_{\mathcal{W}}\ 1)^T$ projects into the image as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \text{CamProj}\left(\bar{E}_{\mathcal{CW}} \boldsymbol{x}_{\mathcal{W}} \right) \tag{16}$$

where $\bar{E}_{\mathcal{CW}}$ is a predicted pose estimate, i.e. the transformation from the world coordinate frame $\mathcal{W}$ to the tablet camera centered frame $\mathcal{C}$. This prediction may be the previous frame's posterior, or could be obtained from the filter described in Section 6. The projection from camera frame to image coordinates is modelled by

$$\text{CamProj}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{bmatrix} \begin{pmatrix} \frac{r'}{r}\frac{x}{z} \\ \frac{r'}{r}\frac{y}{z} \\ 1 \end{pmatrix} \tag{17}$$

with $r = \sqrt{(x/z)^2 + (y/z)^2}$, $r' = r + \alpha r^3 + \beta r^5$ to compensate for radial lens distortion. The relevant parameters for the camera used are known. Figure 1a shows an example video frame captured by the camera with the system's pose estimate rendered over it. In this image, the rendered model is not correctly aligned with the video image: the aim is to compute a camera motion $M_{\mathcal{C}}$ which will properly align the model to give the posterior pose estimate:

$$E_{\mathcal{CW}} = M_{\mathcal{C}} \bar{E}_{\mathcal{CW}} \tag{18}$$

To calculate this motion, sample points are initialised along the visible model edges. From these points, perpendicular searches for the nearest video image edge are performed. This step is illustrated in Figure 1b, in which white lines represent the perpendicular distances to the nearest image edge. This distance measure (which is assumed to be noisy) is written $\hat{d}_i$ for the ith of N sample point. Next, Eq. (16) is differentiated w.r.t. the parameters of the motion $M_{\mathcal{C}}$ to obtain a N×6 Jacobian matrix J such that

$$J_{i,j} = \frac{\partial d_i}{\partial \mu_{\mathcal{C}\text{j}}} = \hat{\boldsymbol{n}}_i \cdot \begin{pmatrix} \frac{\partial u}{\partial \mu_j} \\ \frac{\partial v}{\partial \mu_j} \end{pmatrix} \tag{19}$$

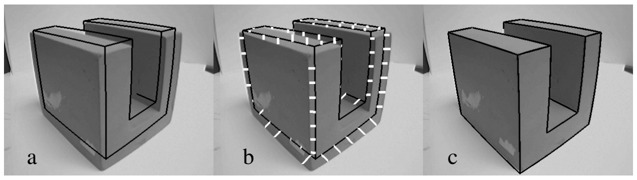

Figure 1. Edge Tracking: a) Prior, b) Measurement c) Posterior

where $\boldsymbol{n}_i$ is the edge normal for the ith sample point. The required motion $M_{\mathcal{C}} = \exp(\boldsymbol{\mu}_{\mathcal{C}})$ may then be found after solution of the equation

$$J \boldsymbol{\mu}_{\mathcal{C}} = \hat{\boldsymbol{d}} \tag{20}$$

which for standard least-squares[2] takes the form

$$\boldsymbol{\mu}_{\mathcal{C}} = J^\dagger \hat{\boldsymbol{d}} = (J^T J)^{-1} J^T \hat{\boldsymbol{d}} \tag{21}$$

An estimate of the accuracy of the motion vector $\boldsymbol{\mu}_{\mathcal{C}}$ is required to use the pose estimates provided by this system in a statistical filter. To calculate this, image measurements are assumed to be corrupted by independent Gaussian noise of 1 pixel standard deviation:

$$\hat{\boldsymbol{d}} = \boldsymbol{d} + \boldsymbol{\delta}\ , \qquad \boldsymbol{\delta} \sim N(\boldsymbol{0}, I_N) \tag{22}$$

Rewriting Eq. (21) in terms of noisy measurements $\hat{\boldsymbol{d}}$ the noisy motion estimate is given by

$$\hat{\boldsymbol{\mu}}_{\mathcal{C}} = J^\dagger \hat{\boldsymbol{d}} \tag{23}$$

The covariance can be found using expectation:

$$\begin{aligned} \Sigma_{\mathcal{C}} &= \text{E}\left[(\hat{\boldsymbol{\mu}}_{\mathcal{C}} - \boldsymbol{\mu}_{\mathcal{C}})(\hat{\boldsymbol{\mu}}_{\mathcal{C}} - \boldsymbol{\mu}_{\mathcal{T}})^T\right] \\ &= \text{E}\left[(J^\dagger \hat{\boldsymbol{d}} - J^\dagger \boldsymbol{d})(J^\dagger \hat{\boldsymbol{d}} - J^\dagger \boldsymbol{d})^T\right] \\ &= \text{E}\left[(J^\dagger \boldsymbol{\delta})(J^\dagger \boldsymbol{\delta})^T\right] \\ &= J^\dagger \text{E}\left[\boldsymbol{\delta}\boldsymbol{\delta}^T\right] J^{\dagger T} \\ &= (J^T J)^{-1} \end{aligned} \tag{24}$$

Our previous work [9] used inertial sensors to deal with the large motions of a head-mounted camera. By contrast, the tablet-mounted camera used here undergoes relatively moderate motion. Further, while both visual and inertial sensors in our previous work produced only relative measurements, the external tracking of Section 5 provides a source of absolute measurements. Occasional failures of the edge-based tracking are therefore tolerable and for this reason (and to save weight and bulk) the inertial sensors described in [9] are not used with the tablet PC.

[2] In practice an M-Estimator is used instead.

5. Outside-in fiducial tracking

The marker-less inside-out tracking described in the previous section depends on an approximate prior pose estimate, and cannot operate if this prior pose is corrupt. To complement this, a second source of pose information should not require information from the past, but calculate a fresh pose estimate at every single frame.

We attach fiducials to the back of the tablet PC and observe these with a camera fixed in the world. Infra-red LEDs are chosen as fiducials as they have a number of advantages over larger printed markers: their small size makes occlusion by users less likely; their size in the image changes insubstantially with distance; finally, using an infra-red filter with the sensor cameras, they are very easy to find in an image and false positive rates (during indoor operation) are negligible.

The disadvantage of using LEDs is that in contrast to paper markers which can have unique patterns printed in them, LEDs cannot be distinguished by appearance. While it is possible to strobe LEDs to determine their identity [21], this requires information to be merged over many frames, whereas we require a full pose estimate each frame. Instead, LEDs are identified based on their relative positions in the image. Six co-planar LEDs are mounted to the back of the tablet with known positions (Fig 2a).

In an offline procedure, four LEDs are selected (Fig 2a) and warped to a unit square (Fig 2b) with a plane-to-plane homography. The warped positions of the remaining two LEDs ($\bigoplus$) form a characteristic of this permutation, which is stored in a table. The table is filled for all 120 permutations for which four LEDs form convex planar shapes.

At runtime, LEDs are detected in a 768x288 greyscale image by thresholding. After removing the effects of lens distortion, four of six LEDs detected in the image are randomly selected (Fig 2c). For each permutation in the stored table, the two characteristic positions from the unit square are warped into the image plane and the error to the remaining detected LEDs is measured (Fig 2d). The permutation with best consensus yields the identity of the detected LEDs. This system copes with the occlusion of a single LED: beyond this no unique identification is possible. While it is possible to establish correspondence using information from past frames or a prior ([2]) this is not attempted here.

After LED identification, tablet frame coordinates of the form $\boldsymbol{x}_\mathcal{T} = (x_\mathcal{T}\ y_\mathcal{T}\ 0\ 1)^T$ of the selected LEDs are known. The image locations are un-projected by the inverse of Eq.(16) to give image plane coordinates of the form $(u'\ v'\ 1)^T$. The plane-to-plane homography H such that

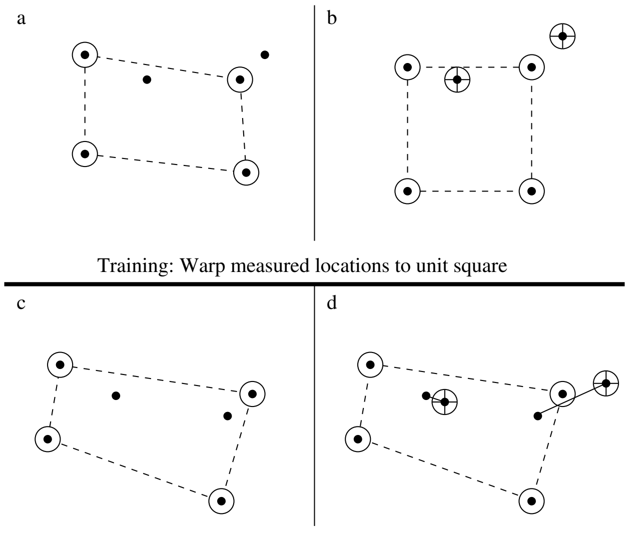
a
b
Training: Warp measured locations to unit square
c
d
Matching: Warp into detected image and measure distance

Figure 2. LED Matching procedure

$$\begin{bmatrix} w'_1 u'_1 & ... & w'_N u'_N \\ w'_1 v'_1 & ... & w'_N v'_N \\ w'_1 & ... & w'_N \end{bmatrix} = H \begin{bmatrix} x_{\mathcal{T}1} & ... & x_{\mathcal{T}N} \\ y_{\mathcal{T}1} & ... & y_{\mathcal{T}N} \\ 1 & ... & 1 \end{bmatrix} \tag{25}$$

is found using standard techniques. From this homography, it is possible to obtain an estimate of the 4×4 transformation matrix $\hat{E}_{\mathcal{ST}}$ which transforms LED coordinates $\boldsymbol{x}_\mathcal{T}$ into the LED camera frame $\mathcal{S}$: such a method is presented e.g. by Kato and Billinghurst [8], and a similar approach is used here.

This pose estimate is then refined for the full projection model and all identified LEDs. Writing the refinement as a small motion $M_\mathcal{T}$,

$$\hat{E}'_{\mathcal{ST}} = \hat{E}_{\mathcal{ST}n} M_\mathcal{T}. \tag{26}$$

The re-projection of the LEDs into the image

$$\begin{pmatrix} \hat{u} \\ \hat{v} \end{pmatrix} = \mathrm{CamProj}\left(\ \hat{E}'_{\mathcal{ST}} \boldsymbol{x}_\mathcal{T}\ \right) \tag{27}$$

can then be differentiated to form a 2N×6 Jacobian matrix J, where

$$J_{2i,j} = \frac{\partial \hat{u}_i}{\partial \mu_{\mathcal{T}j}}, \qquad J_{2i+1,j} = \frac{\partial \hat{v}_i}{\partial \mu_{\mathcal{T}j}}. \tag{28}$$

This matrix is used to minimise the re-projection errors $(\hat{u}-u, \hat{v}-v)$ as in Eq. (23). Analogously to Eq. (24), image locations are assumed corrupted by independent Gaussian noise with $\sigma = 1$ pixel. The covariance in the tablet frame of the final pose estimate is thus

$$\Sigma_\mathcal{T} = (J^T J)^{-1}. \tag{29}$$

6. Filtering

To combine the measurements from the tablet-mounted camera and any fixed cameras observing the LEDs mounted on the back of the tablet, an Extended Kalman filter is employed. This section describes the steps required to filter the two sources of pose information used. To aid readability the notation here is loosely based on Welch and Bishop's excellent tutorial [20] on the filter.

The filter used tracks 12 degrees of freedom: the 6DOF tablet camera pose its 6DOF velocity. However, the filter state x is not stored as a 12-vector. Pose is represented as the transformation matrix $E_{\mathcal{CW}}$ while velocity is stored as a 6-vector $\boldsymbol{v}_{\mathcal{C}}$ in coordinate-frame $\mathcal{C}$. Then, for time t, the filter's estimate $\hat{x}_t$ of the actual system state x_t is

$$\hat{x}_t = \left\{ \hat{E}_{\mathcal{CW}|t},\ \hat{\boldsymbol{v}}_{\mathcal{C}|t} \right\}. \tag{30}$$

The state estimate relates to the true state as

$$\begin{aligned} \hat{E}_{\mathcal{CW}|t} &= \exp(\boldsymbol{\epsilon}_{\text{pose}}) E_{\mathcal{CW}|t} \\ \hat{\boldsymbol{v}}_{\mathcal{C}|t} &= \boldsymbol{\epsilon}_{\text{vel}} + \boldsymbol{v}_{\mathcal{C}|t} \end{aligned} . \tag{31}$$

The state error 12-vector $\boldsymbol{\epsilon}_{\mathcal{C}|t}$ at time t is thus

$$\boldsymbol{\epsilon}_{\mathcal{C}|t} = \begin{pmatrix} \boldsymbol{\epsilon}_{\text{pose}} \\ \boldsymbol{\epsilon}_{\text{vel}} \end{pmatrix} \tag{32}$$

and is modelled as normally distributed

$$\boldsymbol{\epsilon}_{\mathcal{C}|t} \sim N(\mathbf{0}, P_t). \tag{33}$$

where P_t is the filter's state error covariance at time t.

Using a constant-velocity model, the filter's time update equation (sans unknown noise and driving function but parameterised by elapsed time δt) is used to provide a prior estimate of future state:

$$\hat{x}^{-}_{t+\delta t} = f(\hat{x}_t, \delta t) = \left\{ \exp(\hat{\boldsymbol{v}}_{\mathcal{C}|t}\ \delta t) \hat{E}_{\mathcal{CW}|t},\ \hat{\boldsymbol{v}}_{\mathcal{C}|t} \right\}. \tag{34}$$

The corresponding prior state covariance is

$$P^{-}_{t+\delta_t} = A P_t A^T + \sigma_p^2 \left[\begin{array}{cc} 0 & 0 \\ 0 & I_6 \end{array} \right] \tag{35}$$

where σ_p is the system's process noise parameter and A takes the from

$$A = \left[\begin{array}{cc} I_6 & \delta_t I_6 \\ 0 & I_6 \end{array} \right] \tag{36}$$

in accordance with the system dynamics.

To integrate pose measurements from the sensors used, these pose measurements $\hat{E}^{\text{m}}$ are converted to an *innovation motion* which describes the motion from the filter's prior state to the pose described by the measurement:

$$M_{\mathcal{C}} = \hat{E}^{\text{m}}_{\mathcal{CW}} \hat{E}^{-}_{\mathcal{CW}|t+\delta t}{}^{-1} \tag{37}$$

Further, the measurement's covariance $\Sigma^{\text{m}}_{\mathcal{C}}$ is transformed into the filter's reference frame and used to compute the Kalman gain K. Dropping the subscript $t + \delta t$,

$$K = P^{-} H^T \left(H P^{-} H^T + \Sigma^{\text{m}}_{\mathcal{C}} \right)^{-1} \tag{38}$$

where matrix $H = [I_6\ 0]$. The posterior pose is found by weighting the innovation motion by the Kalman gain and applying the result to the prior pose:

$$\hat{E}_{\mathcal{CW}} = \exp\left(K \log(M_{\mathcal{C}})\right) \hat{E}^{-}_{\mathcal{CW}} \tag{39}$$

and the posterior covariance is found as

$$P = (I_6 - KH) P^{-}. \tag{40}$$

Since the inside-out tracking of Section 4 produces pose measurements of the form $\{E_{\mathcal{CW}},\ \Sigma_{\mathcal{C}}\}$ (c.f. Eq. (18,24)), these can be directly filtered as described above. On the other hand, the LED tracking system described in Section 5 produces pose estimates of the form $\{E_{\mathcal{TS}},\ \Sigma_{\mathcal{T}}\}$; for these measurements to be filtered they must first be transformed into the filter's coordinate frame.

For this purpose, knowledge of the transformations $E_{\mathcal{CT}}$ (Tablet camera from tablet back) and $E_{\mathcal{SW}}$ (LED tracking camera from world) is required. Providing the tablet camera is rigidly attached to the tablet and the sensor camera rigidly mounted in the world, these transformations may be considered fixed and need be calibrated only once. While these transformations can not be directly measured, they can be calculated by observing the changes in the two sensor measurements. Chaining together transformations,

$$E_{\mathcal{SW}} = E_{\mathcal{ST}} E_{\mathcal{TC}} E_{\mathcal{CW}}. \tag{41}$$

Inserting an observed motion in the tablet camera frame $M_{\mathcal{C}}$ and a simultaneously observed motion in the tablet back frame $M_{\mathcal{T}}$, this chain remains valid:

$$\begin{array}{rcl} E_{\mathcal{SW}} = E_{\mathcal{ST}} M_{\mathcal{T}} E_{\mathcal{TC}} M_{\mathcal{C}} E_{\mathcal{CW}} & = & E_{\mathcal{ST}} E_{\mathcal{TC}} E_{\mathcal{CW}} \\ M_{\mathcal{T}} E_{\mathcal{TC}} M_{\mathcal{C}} & = & E_{\mathcal{TC}} \\ M_{\mathcal{T}} E_{\mathcal{TC}} & = & E_{\mathcal{TC}} M_{\mathcal{C}}^{-1} \end{array} \tag{42}$$

Baillot *et al* [1] have recently identified this problem as one studied in robotics as $AX = XB$. A closed form solution which can generate an estimate for $E_{\mathcal{TC}}$ from to a minimum of two sets of motion measurements exists [13] and is also used here. Once $E_{\mathcal{TC}}$ (and thus, simultaneously, $E_{\mathcal{SW}}$) has been obtained, measurements from LED tracking can be transformed into the tablet camera frame

$$\begin{aligned} E^{\text{m}}_{\mathcal{CW}} &= E^{-1}_{\mathcal{TC}} E_{\mathcal{TS}} E_{\mathcal{SW}} \\ \Sigma^{\text{m}}_{\mathcal{C}} &= \text{Adj}(E^{-1}_{\mathcal{TC}}) \Sigma_{\mathcal{T}} \text{Adj}(E^{-1}_{\mathcal{TC}})^T \end{aligned} \tag{43}$$

and so measurements from both inside-out and outside-in tracking are accommodated. This completes the equations required for operation of the filter.

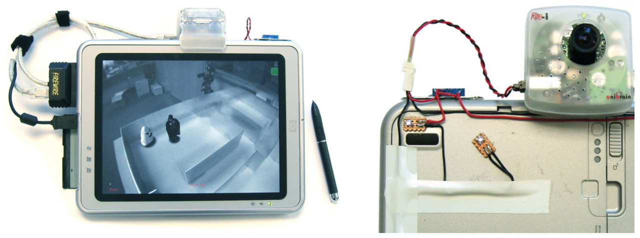

Figure 3. Tablet PC with attached camera, and close-up of two LEDs mounted on back

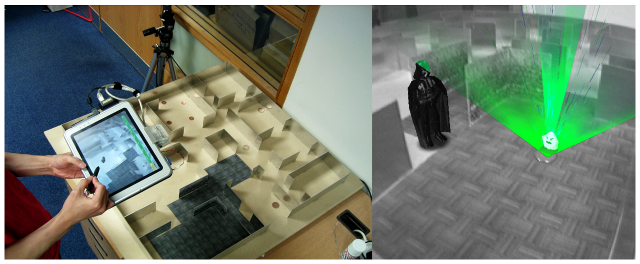

Figure 4. Left, overview of game. Right, players can trap ghosts by throwing coins into the game

7. Tablet-based AR

The edge-based tracking of Section 4 was implemented on an HP Compaq TC1100 tablet PC, pictured in Figure 3. This device uses a 1GHz ULV Pentium-M processor. A 10" screen with 1024x768 pixels is driven by an NVidia Geforce4 420 Go graphics accelerator. Video input is provided by a Fire-i camera fitted with a wide-angle lens. Unfortunately the PCMCIA fire-wire card used provides no power, so the camera is wired up to draw power from the USB port, and this also provides current to the six infra-red LEDs attached to the back of the tablet.

The LEDs are tracked using one or more standard monochrome PAL video cameras operating at field rate (50Hz.) Infra-red transmissive filters are attached to the lenses to block out most of the visible spectrum: The images produced by these cameras are solid black with only the LEDs showing as white dots. The cameras are connected to video capture cards in a standard workstation (2 $\times$ 2.4 GHz.) This machine runs the LED tracking described in Section 5 and the statistical filter of Section 6. Measurements and pose predictions are exchanged between workstation and the tablet PC by wireless network, and the two machine's clocks are synchronisation to sub-millisecond accuracy by aggressive NTP polling.

A prototype entertainment application is in development to evaluate tracking performance. This application is centered around a real-world 0.8$\times$0.8m playing field resembling a large "Cluedo" board. Figure 4 shows the playing field, the tablet PC and one of the LED-tracking cameras. The field is divided into rooms by thin vertical walls: the 3D locations of these walls are known and form the 3D model which the edge-based tracking system requires.

The player guides a virtual character around the real world using the tablet's pen to point at target positions. The aim of the game is to collect items from rooms while avoiding contact with virtual ghosts controlled by the computer. Ghosts can be removed from the game by throwing *ghost traps* into their path: The player achieves this by physically throwing coins into the playing field, whereupon ghosts (and unfortunate players) in proximity of the landed coin are sucked into a virtual vortex. Coins are detected using a ceiling-mounted camera: this is connected to the workstation which employs background subtraction and an ad-hoc circle detector. The image location of the detected coin is mapped into the playing field and transmitted to the tablet PC which runs the game logic.

To provide augmented visuals, the tablet PC operates a track-render loop. For each video frame received by the camera, tracking as described in Section 4 is performed. This yields a pose estimate $E_{\mathcal{CW}}$ which is used to render the augmented visuals. Rendering is performed using accelerated OpenGL. However, since a wide-angle lens and projection model with radial distortion terms (Eq. (17)) are used, the augmented graphics cannot be rendered directly into the frame-buffer: instead, the texture-mapping approach of Watson and Hodges [19] is employed, whereby geometry is first rendered without radial distortion, and the rendered image is subsequently warped using a textured grid.

For video see-through, this requires a GL visual with destination alpha support. Each frame, the z-buffer is cleared and the frame-buffer set transparent. Next, the geometry used for edge tracking is rendered without distortion into the z-buffer only. This serves to correctly occlude the augmented visuals, which are rendered without distortion into the z and colour buffers. The framebuffer, now containing only the augmented visuals (player, ghosts etc.) over a transparent background, is copied into a texture map. The framebuffer is then overwritten with the video camera image. Finally, the just generated texture map is drawn over the video image using a 20x20 quad mesh (whose vertices have been distorted in accordance with the lens distortion) to form the final composited scene.

8. Occlusion refinement

In augmented reality, virtual objects are often placed in the real world. To appear believable, these objects should not only be well-registered with the real world, they should also occlude real objects behind them, and be occluded by real objects in front. The accuracy of this occlusion greatly affects the user's perception that the virtual object belongs in the scene. If too much is occluded the virtual object looks like a cardboard cut-out. If too little is occluded, this destroys the depth cues.

This occlusion is often resolved by z-buffering: by populating the z-buffer with an estimate of the real world's depth, occlusion of the subsequently rendered virtual objects is automatically handled by the rendering hardware. The z-buffer can be filled with information generated from a stored 3D model and a pose estimate (c.f. [4] and the approach in the previous section) or data generated on-line (using e.g. depth from stereo [22]). Whichever method is used, the values in the z-buffer will occasionally not correspond to the real depth in the scene, and occlusion errors will occur.

Considering only those systems which assume knowledge of the occluding real geometry, the most obvious source of error is an inaccurate model of this geometry. However, even if the model is accurate, tracking errors (or jitter) or incorrect projection parameters can produce noticeable occlusion errors in the image. This is particularly true of systems in which the tracked feature and the occluding geometry are some image distance apart: in this case, any small rotational tracking error produces an amplified occlusion error. By tracking the visible edges in the scene to obtain pose, our system is also optimising for those features which cause occlusion and this goes a long way to providing a good level of realism. However, the system still produces substantial occlusion errors on occasion. Figure 5a shows an example in which the occluding wall has been rendered into the z-buffer too far to the right, so that too much of the virtual character is occluded. Even though the position of the occluding wall was measured during tracking, it is drawn in the position which is best fits *all* measured edges. To solve this problem, an approach has been developed which optimises the modelled location of an occluding edge using measurements from that edge only. To achieve this, the rendering process described in the previous section must be modified:

As before, the frame-buffer is cleared to be transparent. However the z-buffer is no longer populated by rendering the entire occluding model; instead, virtual geometry is occluded object by object. For each object, the potentially occluding world geometry is identified. This task is simplified by the structure of the world model used, which contains only vertical walls. Once an occluding wall has been identified, it is projected to a *clipping polygon* in the image plane. The object-occluding sides of this polygon are then optimised by searching in the video image. Figure 5b shows this optimisation, as well as the non-occluding edges of the clipping polygon drawn as solid lines.

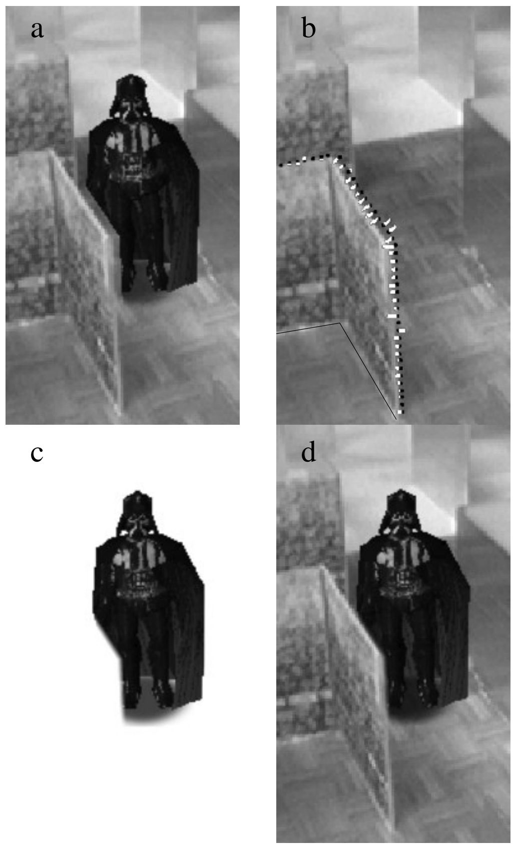

Figure 5. Occlusion refinement. a) No occlusion refinement: too much of the virtual object is clipped, b) clipping polygon is refined through image search, c) character is rendered and clipped with refined polygon, d) composited scene.

At this point, clipping the virtual object could simply be performed by rendering the clipping polygon into the stencil buffer and using this as a mask for rendering the virtual object. However this would produce a very sharp edge, whereas edges in the video image are slightly blurred by the camera optics. This produces an unrealistic and jarring contrast. It is preferable to clip the virtual object with a slightly graduated edge to blend it into the blurred video edge.

To achieve this blending effect it is necessary to first render the object unclipped, and to subsequently overwrite its alpha channel with a modified clipping polygon. The clipping polygon has the occluding edges transformed into thin quads which are transparent along one side and opaque on the other. OpenGL interpolates this into a smooth alpha-graduated occluding edge. The stencil buffer can be used to ensure that the clipping polygon only overwrites pixels which were drawn by the corresponding virtual object.

Figure 5c shows a clipped virtual object. The final scene is composited as described in Section 7, and is pictured in Figure 5d, showing substantial improvement over the composite shown in Figure 5a.

9. Results

9.1. Real-time performance

The full AR system described in Sections 7 and 8 operates at between 25 and 30 frames per second on the tablet. CPU usage on the tablet registers at 60%. We expect to achieve consistent 30 frames per second performance with some optimisation of graphics calls. On the workstation, LED tracking, filtering and coin detection run at full framerate and occupy a total of 15% of processor time.

9.2. Errors

Tracking jitter was evaluated by keeping the tablet static and observing the noise in incoming measurements. Typical RMS jitter values for the edge-based and LED tracking are tabulated in Table 1. Tracking jitter reduces the apparent registration of real and virtual objects, with virtual objects appearing to wobble on a static background.

	Edges		LEDS	
	Trans/	Rot	Trans/	Rot
Jitter (mm/deg)	1.1	0.17	5	0.5
σ (mm/deg)	1.0	0.15	8	2.85

Table 1. Tracking jitter compared to estimated standard deviation

The observed jitter of the edge-based tracking agrees with the expected error. The LED measurements yield lower observed jitter than the expected error. This is likely to be due to the fact that the LED centroids in the image can be extracted to sub-pixel accuracy and the σ=1 pixel assumption in Section 5 is overly pessimistic.

A systematic error between LED measurements and tablet camera measurements was observed in some situations. Depending on the position of the tablet in the playing volume, this error was as large as 2cm. It is likely that this errors is caused by inaccuracies in the calibration of $E_{\mathcal{SW}}$ and $E_{\mathcal{TC}}$ and errors in camera parameter values.

9.3. Dynamic performance

A video file demonstrating tracking performance is enclosed. In this video, the tracked edges are rendered into the scene to allow visual inspection of tracking performance. During normal operation, these edges are not rendered.

In standalone operation, the edge based tracking of the tablet camera is prone to failure on rapid motions. Further, there is a possibility of the edge based tracking falling into local minima. These failure mechanisms are illustrated in the enclosed video file.

The LED tracking does not suffer any adverse effects from rapid motion. The LEDs are bright enough that the LED camera's exposure time can be set to a small enough value to eliminate motion blur. However, the LED tracking by itself is not accurate enough to properly register the augmented visuals. This is due to both the systematic pose errors described above and the relatively large tracking jitter.

When edge-based tracking and LED tracking are combined, the LED tracking's initialisation is for the most part sufficient to allow the edge-based tracking to converge. Recovery from total edge-tracking failure is possible as long as the LEDs are in view of the observing camera. It should be noted this volume is larger than appears in the result video, in which the tablet is tethered for filming.

The systematic error described above can produce oscillatory behaviour in the system state. However, since the augmented visuals are rendered using the edge tracking posterior, this oscillatory behaviour of the state is not observable in the AR display - there is however a small probability that at any given frame, edge-tracking will not converge correctly, and this causes occasional one-frame glitches in the display. Initial attempts to adaptively weight [5] the innovations of the different sensors in the EKF have shown great potential in reducing this oscillation. Simultaneoulsy, re-convergence after edge-tracking failure can be sped up. This is an area of future work.

9.4. Occlusion Refinement

The accompanying video file demonstrates the effect of occlusion refinement. In most scenes, the use of occlusion refinement is beneficial to the composited appearance. However for some configurations the refinement introduces new errors. In particular this is the case if an occluding edge in the image is very low-contrast and in close proximity to other texture edges or shadows. In this case the edge position refinement converges on an incorrect edge, producing large, often dynamic, and mostly very noticeable occlusion errors. Some preliminary efforts to reduce these errors (the use of M-estimation, and the imposition of zero-motion priors on an edge's displacement and rotation) have reduced the likelihood of such errors, but there remains scope for improvement (by e.g. using alternative line-fitting techniques, information from previous frames or adjoining edges.)

In the absence of correspondence failures, the occlusion refinement system enhances the appearance of the composited scene. In particular, the visible effect of tracking jitter can be reduced. Further, the "crawling jaggies" effect when occluding edges are near-horizontal or near-vertical is mostly eliminated by the alpha-blended clipping procedure.

9.5. Conclusion

This paper has demonstrated the feasibility of robust and accurate AR on a tablet PC without the use of markers placed in the scene. The complementary strengths of inside-out edge-based tracking and outside-in LED tracking are combined in an Extended Kalman Filter, using a powerful mathematical framework which facilitates the manipulation of noisy data from different reference frames.

Further, this paper has shown that by fine-tuning individual occluding edges in the image, the apparent registration of virtual objects in the real world can be greatly improved.

In direct comparison to PDAs, the tablet (1.4kg) is uncomfortably heavy to hold one-handed for extended periods of time. Further, the device becomes rather hot. It is expected that as tablet PCs become lighter and PDAs more powerful, full frame-rate handheld AR will become a very practical possibility.

References

[1] T. Baillot, S. Julier, D. Brown, and M. Livingston. A tracker alignment framework for augmented reality. In *Proc. Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 142–150, Tokyo, October 2003.

[2] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–61, 1995.

[3] M. O. Berger. Resolving occlusion in augmented reality: a contour based approach without 3d reconstruction. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 91. IEEE Computer Society, 1997.

[4] D. Breen, R. Whitaker, E. Rose, and M. Tuceryan. Interactive occlusion and automatic object placement for augmented reality. In *Proc. of Eurographics*, pages 11–22, Poitiers, France, August 1996.

[5] T. Cipra and R. Romera. Robust kalman filter and its application in time series analysis. *Kybernetika*, 27(6):481–494, 1991.

[6] A. Fuhrmann, G. Hesina, F. Faure, and M. Gervautz. Occlusion in collaborative augmented environments. In *Proc. 5th EUROGRAPHICS Workshop on Virtual Environments*, Vienna, June 1999.

[7] M. Kanbara, T. Okuma, H. Takemura, and N. Yokoya. A stereoscopic video see-through augmented reality system based on real-time vision-based registration. In *Proc. IEEE Virtual Reality 2000 (VR2000)*, pages 255–262, March 2000.

[8] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proc. 2nd Int'l Workshop on Augmented Reality*, pages 85–94, San Francisco, CA, Oct 1999.

[9] G. Klein and T. Drummond. Robust visual tracking for non-instrumented augmented reality. In *Proc. Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 113–122, October 2003.

[10] V. Lepetit and M. O. Berger. Handling occlusions in augmented reality systems: A semi-automatic method. In *Proc. International Symposium on Augmented Reality (ISAR)*, pages 197–146, October 2000.

[11] A. MacWilliams, C. Sandor, M. Wagner, M. Bauer, G. Klinker, and B. Brügge. Herding sheep: Live system development for distributed augmented reality. In *Proc. Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, Tokyo, October 2003.

[12] C. Owen, F. Xiao, and P. Middlin. What is the best fiducial? In *Proc. First IEEE International Augmented Reality Toolkit Workshop*, pages 98–105, Darmstadt, September 2002.

[13] F. Park and B. Martin. Robot sensor calibration: Solving AX=XB on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, 1994.

[14] K. Satoh, S. Uchiyama, H. Yamamoto, and H. Tamura. Robust vision-based registration utilizing bird's-eye view with user's view. In *Proc. Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, Tokyo, October 2003.

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

[16] V. Vlahakis, N. Ioannidis, J. Karigiannis, M. Tsotros, and M. Gounaris. Virtual reality and information technology for archaeological site promotion. In *Proc. 5th International Conference on Business Information Systems (BIS02)*, Poznan, Poland, April 2002.

[17] D. Wagner and I. Barakonyi. Augmented reality kanji learning. In *Proc. Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, Tokyo, October 2003.

[18] D. Wagner and D. Schmalstieg. Artoolkit on the pocketpc platform. Technical Report TR-188-2-2003-23, Technical University of Vienna, 2003.

[19] 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, March 1995.

[20] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, 1995. Updated 2002.

[21] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. The hiball tracker: High-performance wide-area tracking for virtual and augmented environments. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, 1999.

[22] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *Proc. Symposium on Interactive 3D Graphics*, pages 5–12, New York, April 1995.

[23] W. Zhu, C. Owen, H. Li, and J.-H. Lee. Personalized in-store e-commerce with the promopad: an augmented reality shopping assistant. *Electronic Journal for E-commerce Tools and Applications*, 1(3), Feb 2004.