

A Method for Calibrating See-through Head-mounted Displays for AR *

Erin McGarrity
Mihran Tuceryan

Department of Computer and Information Science
Indiana University Purdue University Indianapolis (IUPUI)
Indianapolis, IN 46202-5132
emcgarr@cs.iupui.edu
tuceryan@acm.org

Abstract

In order to have a working AR system, the see-through system must be calibrated such that the internal models of objects match their physical counterparts. By match, we mean they should have the same position, orientation, and size information as well as any intrinsic parameters (such as focal lengths in the case of cameras) that their physical counterparts have. To this end, a procedure must be developed which estimates the parameters of these internal models. This calibration method must be both accurate and simple to use.

This paper reports on our efforts to implement a calibration method for a see-through head-mounted display. We use a dynamic system in which a user interactively modifies the camera parameters until the image of a calibration object matches the image of a corresponding physical object. The calibration method is dynamic in the sense that we do not require the user's head to be immobilized.

1. Introduction

In order for augmented reality to be effective the real and computer-generated objects must be accurately positioned relative to each other and properties of certain devices must be accurately specified. This implies that certain measurements or *calibrations* need to be made at the start of the system. These calibrations involve measuring the pose of various components such as the trackers, pointers, cameras, etc. What needs to be calibrated in an AR system and how easy or difficult it is to accomplish this depends on the architecture of the particular system and what types of components are used.

*Published in the Proceedings of the IEEE 2nd International Workshop on Augmented Reality (IWAR 99), San Francisco, CA, Oct 20-21, 1999.

A detailed view of some of the past work that we have studied on calibration can be found in [3, 5, 7, 9, 12, 13]. In [1] Azuma classifies AR systems according to the amount of immersion and interaction. Kutulakos et al. propose an uncalibrated AR system in [8]. Azuma et al. [2], and Holloway [6] have studied registration errors in HMDs.

There are two major modes of display which determine what types of technical problems arise in augmented reality systems, what the system architecture is, and how these problems are to be solved: (i) video-see-through systems and (ii) optical see-through systems. The calibration issues in a video-see-through system is described in detail elsewhere [12]. We define an optical see-through system as the combination of a see-through head-mounted display and a human eye.

In this paper, we look at the calibration issues in an AR system of the second type, namely, an optical see-through system. In particular, we concentrate on the camera calibration in an optical see-through system and describe a method of calibration in such a system. We also apply this method to calibrate a stereo head mounted system. In Section 2 we give a brief survey of prior work. In Section 3, we describe the hardware and software architecture of our system. In Section 4, we present an overview of the calibration requirements in an optical-see-through AR system. In Section 5, we present the details of our camera calibration approach, followed by, in Section 6, a description and discussion of experimental results. Finally, in Section 7, we give our conclusions.

2. Overview of the hardware and software

The typical optical see-through AR system hardware is illustrated in Figure 2. In this configuration, the display consists of a pair of *i-glasses*TM head-mounted display (HMD) which can be used both as immersive displays as well as

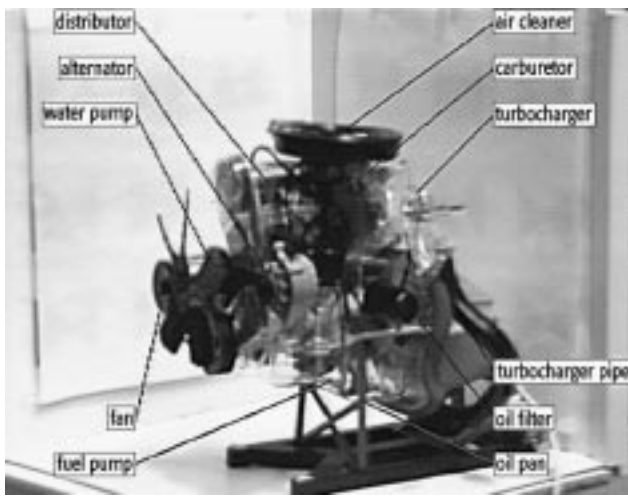


Figure 1. An example application of augmented reality as a tool to assist a technician in a mechanical repair task: the parts of a complex engine being labeled.

see-through displays by removing a piece of opaque plastic from the front of the display screens. Since our research involves augmented reality systems, we have been using these HMD's as see-through displays permanently. The graphical image is generated by the workstation hardware and displayed on the workstation's monitor which is fed at the same time to the *i-glasses*TM over a VGA port. A 6-degrees-of-freedom (6-DOF) magnetic tracker, which is capable of sensing the three translational and the three rotational degrees of freedom, provides the workstation with continually updated values for the position and orientation of the tracked objects which includes the *i-glasses*TM and a 3D mouse pointing device.

The software is based on the Grasp system that was developed at ECRC for the purposes of writing AR applications. We have added the calibration capabilities to the Grasp software and tested our methods in this environment. The Grasp software was implemented using the C++ programming language.

3. Overview of calibration requirements

In an AR system there are both "real" entities in the user's environment and virtual entities. Calibration is the process of instantiating parameter values for "models"¹ which map the physical environment to internal representations, so that the computer's internal model matches the

¹By model, we mean here a mathematical model such as the pinhole camera, and not a geometric model.

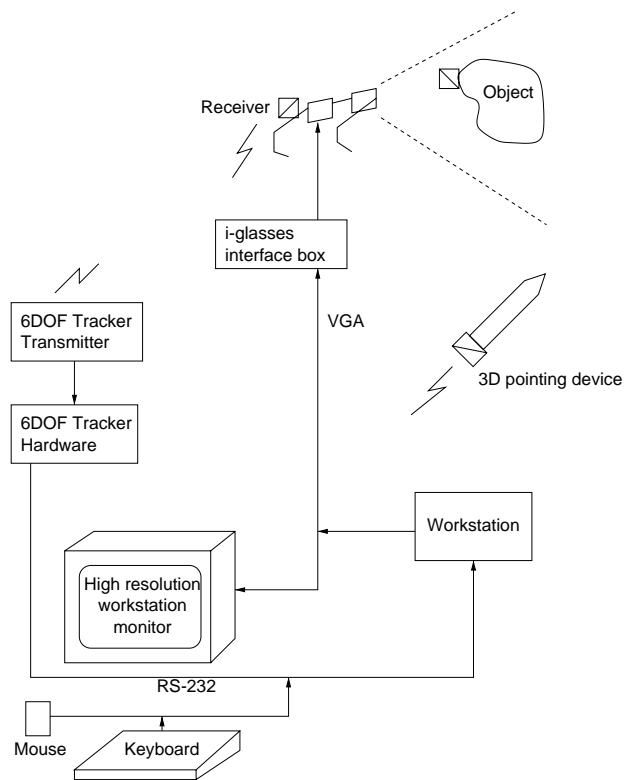


Figure 2. The hardware diagram of a typical see-through augmented reality system. The see-through displays we use are from *i-glasses*TM, and have a limited resolution (640 × 480).

physical world. These models may be the optical characteristics of a physical camera as well as position and orientation (pose) information of various entities such as the camera, the magnetic trackers, and the various objects.

For an AR system to be successful it is crucial that this calibration process be both complete and accurate. Otherwise, the scene rendered by the computer using the internal model of the world will look unrealistic. For example, objects rendered by the computer using a virtual camera whose intrinsic parameters did not match those of the real camera would result in unrealistic and distorted images which looked out of place compared to the physical world.

The calibration requirements of a video-see-through augmented reality system have been described elsewhere [12]. We briefly summarize the highlights of this system as modified for an optical see-through system to determine its calibration requirements. Figure 3 shows all the local coordinate systems and their relationships in a typical optical-see-through AR system. All the calibration requirements for such a system originate from the fact that all the trans-

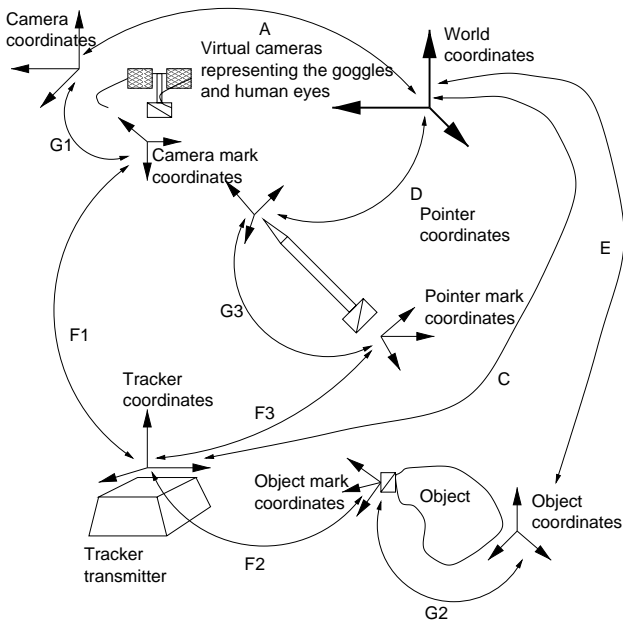


Figure 3. Grasp coordinate systems and their relationships.

formations shown must be known during the operation of the AR system. Some of these transformations are directly read from sensors such as the magnetic trackers. However, some of them need to be estimated through a calibration process and some of them inferred and computed from the rest of the transformations.

The coordinate systems are related to each other by a set of rigid transformations. The central reference is the **World Coordinate System** which is at a fixed and known location relative to the operating environment. During the operation of an AR system, all of the components need to operate in a unified framework which in the case of the Grasp system is the world coordinate system.

The main calibration steps are the following:

1. camera calibration (transformation **A** and intrinsic camera parameters).
2. pointer calibration (transformation G_3),
3. tracker transmitter calibration (transformation **C**),
4. object calibration (transformation **E**),
5. tracker mark calibration, one per tracked entity (transformations G_i). A *mark* is a tracker receiver that is attached to an object being tracked.

Camera Calibration is the process by which the extrinsic camera parameters (location and orientation) as well as

the intrinsic camera parameters (focal length, image center, and aspect ratio) are calculated. This process calculates the transformation labeled **A** in Figure 3 as well as the camera intrinsic parameters. In the case of a video-see-through camera calibration system, this would be the estimation of the parameters for the physical camera. In the case of optical see-through AR system, we would estimate the parameters of the virtual camera which models the combined display system formed by the *i-glasses*TM display and the human visual system. In the latter case, unlike the video-see-through display, there is no explicit image of the real world coming from a camera from which measurements can be made and that can be combined with the computer generated graphics. Therefore, we need to define and design the calibration procedure for an optical see-through system differently from the traditional camera calibration systems in which we have access to the image.

Pointer Calibration is used to calculate the geometry of the pointer object used by the Grasp system applications. In particular, this calibration step calculates the position of the tip of the pointer relative to the tracker mark, i.e., the transformation between the coordinate system of the pointer object and the coordinate system of the tracker mark (the transformation labeled G_3). This step is necessary before many of the subsequent calibration steps can be completed as they require the use of the pointer by the user to pick points on real objects. Transformation G_3 can be estimated once and stored and would be valid as long as the rigid attachment of the receiver does not change.

Tracker Transmitter Calibration calculates the position and orientation of the tracker's coordinate system within the world coordinate system (the transformation represented by the arc labeled **C** in Figure 3). This transformation is calculated indirectly after calculating the transformations **D** and G_3 and measuring transformation F_3 .

The details of tracker, pointer, and video camera calibrations are described in [12]. In this paper we focus on the camera calibration for the optical-see-through display system.

4. Camera calibration for see-through displays

This section details the necessary camera calibration steps for the see-through head-mounted display. The camera calibration method described in our previous work was based on using the relationship between the projected image positions of known 3D points and their 3D positions. From this well known mathematical relationship, the camera parameters were estimated [12]. This assumes that we have access to the picture points (pixel) which we can select and whose image coordinates we can obtain. This can be done in a video-see-through display system because we can always access the image digitized by the video camera

and use it to analyze the input images. With a see-through system, the images of the scene are formed on the retina of the human user's eye and we do not have direct access to the image pixels. Therefore, we need to have a different approach to the calibration. The approach described in this paper uses a dynamic process in the forward direction (i.e., from 3D objects to 2D projected images) and let the user interactively adjust (estimate) the parameters of the imaging system until the projected image of the calibration model as seen by the human eye matches the image of the real calibration object in the scene. *This is a truly dynamic system in the sense that while the user is interactively adjusting the camera parameters to align the displayed image, he is free to move his head.* That is, there is no requirement on the user to stay still or to keep his head in a static position. The system updates the graphics reflecting the changes in the transformation \mathbf{F}_1 , read by the camera marker. The parameters estimated by this calibration procedure are the intrinsic camera parameters and the camera-to-mark transformation \mathbf{G}_1 . The transformation \mathbf{A} then can be obtained from the transformation \mathbf{G}_1 (estimated from camera calibration), from \mathbf{F}_1 (read by the tracker), and from \mathbf{C} (estimated by the tracker transmitter calibration).

In the following sections, we first briefly describe the camera model we are using which defines the parameters to be estimated. We then describe the dynamic estimation process. The calibration procedure is based on the through-the-camera interaction described in a paper by Gleicher and Witkin [4]. We give a quaternion based camera transformation model which reduces the redundant degrees of freedom by reducing the number of parameters to be estimated. This is then followed by a description of applying the current calibration methods to a stereo see-through display system.

4.1. Camera model

A simple pinhole model (Figure 4) is used for the camera, which defines the basic projective imaging geometry with which the 3D objects are projected onto the 2D image surface. This is an ideal model commonly used in computer graphics and computer vision to capture the imaging geometry. It does not account for certain optical effects (such as non-linear distortions) that are often properties of real cameras. There are different ways of setting up the coordinate systems, and in our model we use a right-handed coordinate system in which the center of projection is at the origin and the image plane is at a distance f (focal length) away from it.

The image of a point in 3D is formed by passing a ray through the point and the center of projection, O_C , and then by computing the intersection of this ray with the image plane. The equations for this case are obtained by using

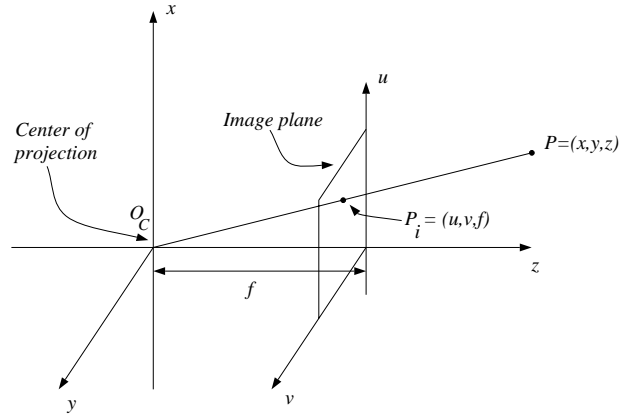


Figure 4. The geometry of the simple pinhole camera model for perspective transformation.

similar triangles,

$$u = fx/z \quad \text{and} \quad v = fy/z. \quad (1)$$

In addition, the camera model used for the generation of the graphical images and for the calibration of the camera has to work with the pixel coordinate system on the display device. This pixel coordinate system has to be accounted for in the mathematical model. The relationships between the screen coordinates, the pinhole model, and the world coordinates is shown in Figure 5 which is the model used for the camera calibration procedure described below.

The position and orientation (pose) of the camera with respect to the world coordinate system $O(x, y, z)$ is defined by a rigid transformation

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \mathbf{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \mathbf{T}, \quad (2)$$

where \mathbf{R} is a 3×3 rotation matrix $[r_{ij}]$, and \mathbf{T} is a translation vector, $[t_x, t_y, t_z]^T$.

The pixel coordinates are related to the image plane coordinates by the following equations (r stands for image row and c for image column):

$$\begin{aligned} r - r_0 &= s_u u, \\ c - c_0 &= s_v v, \end{aligned} \quad (3)$$

where (r_0, c_0) are the pixel coordinates of the image plane coordinate system O' . s_u and s_v are needed in order to (i) account for the proper sign (note that the r -axis and u -axis are pointing in opposite directions), and (ii) the non-isotropic nature of some cameras (i.e., cameras with non-square pixels) and thus also capture the aspect ratio of the

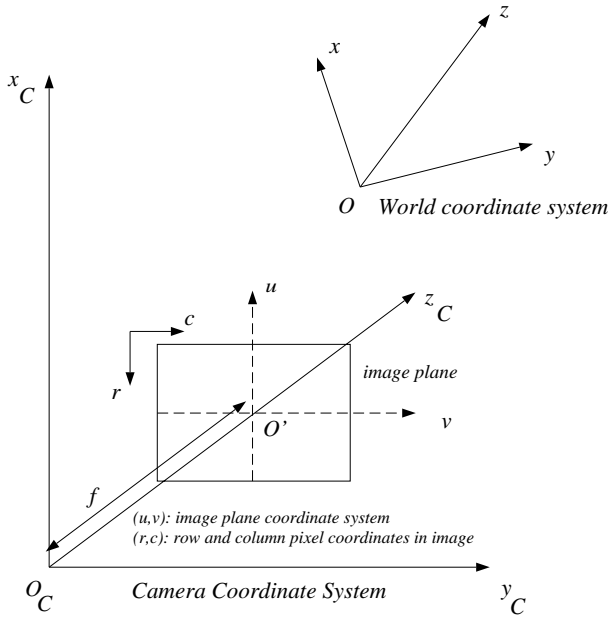


Figure 5. The various coordinate systems with respect to each other.

camera. Let $f_u = s_u f$ and $f_v = s_v f$. Then the four quantities f_u , f_v , r_0 and c_0 are called the *intrinsic camera parameters*. The transformations \mathbf{R} and \mathbf{T} are known as the *extrinsic camera parameters*.

Substituting all the expressions in Equations 2 and 3 into Equation 1, we get

$$\begin{aligned} \frac{u}{f} &= \frac{r - r_0}{s_u f} = \frac{r - r_0}{f_u} = \frac{r_{11}x + r_{12}y + r_{13}z + t_x}{r_{31}x + r_{32}y + r_{33}z + t_z}, \\ \frac{v}{f} &= \frac{c - c_0}{s_v f} = \frac{c - c_0}{f_v} = \frac{r_{21}x + r_{22}y + r_{23}z + t_y}{r_{31}x + r_{32}y + r_{33}z + t_z}. \end{aligned} \quad (4)$$

For this application, we use a quaternion [10] to represent the rotation transformation of the camera to reduce the number of redundant parameters which would be introduced by a 3×3 rotation matrix. The details of this camera are given in the next section.

4.2. Quaternion camera control

Let ν be the vector which describes the camera. Using a quaternion for the rotation transformation, we can write it as

$$\nu = [t_x, t_y, t_z, q_x, q_y, q_z, q_w, f_u, f_v, r_0, c_0]^T, \quad (5)$$

where t_i , $i \in \{x, y, z\}$ describes the translation, q_j , $j \in \{x, y, z, w\}$ describes the quaternion rotation, f_u and f_v are

the focal lengths and r_0 and c_0 are the coordinates of the image center.

In order to render our scene we will need to construct homogeneous transformation matrices from ν . We will use three matrices, one for rotation, one for translation and one for projection. These matrices are given in Equations 6, 8, and 9.

To convert the quaternion into a rotation matrix we use

$$\mathbf{Q} = \frac{2}{|q^2|} \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & 0 \\ Q_{21} & Q_{22} & Q_{23} & 0 \\ Q_{31} & Q_{32} & Q_{33} & 0 \\ 0 & 0 & 0 & Q_{44} \end{pmatrix}, \quad (6)$$

where,

$$\begin{aligned} Q_{11} &= \frac{|q|^2}{2} - q_y^2 - q_z^2, & Q_{12} &= q_x q_y + q_w q_z, \\ Q_{13} &= q_x q_z - q_w q_y, & Q_{21} &= q_x q_y - q_w q_z, \\ Q_{22} &= \frac{|q|^2}{2} - q_x^2 - q_z^2, & Q_{23} &= q_w q_x + q_y q_z, \\ Q_{31} &= q_w q_y + q_x q_z, & Q_{32} &= q_y q_z - q_w q_x, \\ Q_{33} &= \frac{|q|^2}{2} - q_x^2 - q_y^2, & Q_{44} &= \frac{|q|^2}{2}. \end{aligned} \quad (7)$$

The translation matrix for the camera is given by

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (8)$$

The projection matrix is based on the focal lengths and image centers as follows

$$\mathbf{P} = \begin{pmatrix} f_u & 0 & r_0 & 0 \\ 0 & f_v & c_0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (9)$$

The concatenation of matrices in Equations 6, 8, and 9 produces the world to camera viewing transformation

$$\mathbf{M} = \mathbf{P}\mathbf{T}\mathbf{Q}. \quad (10)$$

In our system, the above transformation is actually used to compute the transformation from the marker coordinates to the display coordinates (transformation \mathbf{G}_1 in Figure 3). We can write the world-to-display transformation as

$$\mathbf{V} = \mathbf{M}\mathbf{F}_1\mathbf{C}, \quad (11)$$

where \mathbf{F}_1 is the camera mark transformation which is read from the tracker at each time-step and \mathbf{C} is the world-to-tracker transformation which is obtained from tracker calibration. At each time-step, we recompute \mathbf{V} and update the graphics display.

4.3. Interactive framework for the dynamic parameter estimation

The process of calibrating the camera using this approach consists of moving landmark points in the 2D image by grabbing and dragging them until they are aligned with their corresponding physical points in the image. During the dragging process, at every time interval, a set of dynamic equations described below are solved for the camera parameters and the resulting projected image of the dragged model is displayed. This process continues until a sufficient number of points have been aligned with their physical counterparts so that the entire calibration object model is aligned with the physical calibration object.

We have found that the user interaction is very difficult when trying to solve for all the parameters at once. Therefore, we have broken the calibration procedure into a series of moded interactions in which the user can separately translate, rotate, and scale (by varying the focal length parameters) the virtual camera. These modes may be selected as the user desires, with the goal being to align the virtual object to the physical object.

4.4. Numerical estimation of the camera parameters

The heart of the calibration procedure in this case comes from a modification of the method described by Gleicher and Witkin [4]. The basic framework of this approach starts with the imaging equation

$$\mathbf{p} = \mathbf{h}(\mathbf{V}\mathbf{x}), \quad (12)$$

where \mathbf{x} is the 3D coordinate vector of a world point, \mathbf{V} is a homogeneous matrix representing the combined projection and viewing transforms as described above, and \mathbf{h} is a function that converts a point in homogeneous coordinates to Euclidean coordinates. In particular, let $\mathbf{x} = [x, y, z, w]^T$ be the homogeneous coordinates of a point. It follows that

$$\mathbf{h}(\mathbf{x}) = \begin{pmatrix} x/w \\ y/w \end{pmatrix} \quad (13)$$

is the 2D projected point of \mathbf{x} .

Normally, one would try to estimate the camera parameters (extrinsic and intrinsic) such that the error is minimized in the image between projected points of a known object (calibration object) using the estimated parameters and the image of the physical object. This is exactly what was done in our previous work with camera calibration for a video-see-through system. In the current method, the approach is to let the user adjust the parameters so that a rendered image of the calibration image is matched with the image of the object. This is done dynamically and the user sees the

result of his adjustments in real time. The dynamic framework for this is done by working with and estimating the time derivatives of the 2D image points that, in turn, depend on the changes in the camera parameters. To see this, we differentiate the point in Equation 12 to obtain

$$\dot{\mathbf{p}} = \mathbf{h}'(\mathbf{V}\mathbf{x}) \left(\frac{\partial(\mathbf{V}\mathbf{x})}{\partial \boldsymbol{\nu}} \right) \dot{\boldsymbol{\nu}}. \quad (14)$$

If we let

$$\mathbf{J} = \mathbf{h}'(\mathbf{V}\mathbf{x}) \left(\frac{\partial(\mathbf{V}\mathbf{x})}{\partial \boldsymbol{\nu}} \right), \quad (15)$$

then we obtain the linear equation,

$$\dot{\mathbf{p}} = \mathbf{J}\dot{\boldsymbol{\nu}}, \quad (16)$$

to solve in terms of the rates of changes of the parameters. Once the $\dot{\boldsymbol{\nu}}$ vector is estimated, we can integrate over time to obtain the parameters $\boldsymbol{\nu}$.

4.5. Controlling the image points

Equation 16 relates the rate of change in parameters to the rate of change in the position of the point.

In the calibration process, we allow the user to change the positions of landmark calibration points interactively by dragging them. We then use the new positions of these points estimate the change in the camera parameters. Equation 16 is the main engine for doing this. We now give the details of this as used in our calibration method. We first focus on controlling a single point. Assume that the user somehow specifies an image point velocity $\dot{\mathbf{p}}_0$. Since the \mathbf{J} matrix in Equation 16 is not square, we cannot solve for the $\dot{\boldsymbol{\nu}}$ directly. What we would like to do is to minimize the error between $\dot{\mathbf{p}}$ and a specified value for $\dot{\mathbf{p}}_0$ such that $\dot{\mathbf{p}} = \dot{\mathbf{p}}_0$. That is, solve

$$E = \frac{(\dot{\boldsymbol{\nu}} - \dot{\boldsymbol{\nu}}_0) \cdot (\dot{\boldsymbol{\nu}} - \dot{\boldsymbol{\nu}}_0)}{2} \quad (17)$$

subject to the constraint $\dot{\mathbf{p}} - \dot{\mathbf{p}}_0 = 0$. The solution to this equation is described in great detail in [4]. A summary of the solution follows.

First, we enforce the constraint $\dot{\mathbf{p}} - \dot{\mathbf{p}}_0 = 0$ by writing

$$\dot{\mathbf{p}}_0 = \mathbf{J}\dot{\boldsymbol{\nu}}, \quad (18)$$

which provides the solution with the minimal change in $\dot{\boldsymbol{\nu}}$. We enforce the constraints by using the Lagrange multiplier

$$\frac{dE}{d\dot{\boldsymbol{\nu}}} = \dot{\boldsymbol{\nu}} - \dot{\boldsymbol{\nu}}_0 = \mathbf{J}^T \lambda. \quad (19)$$

Multiplying equation 19 by \mathbf{J} and substituting equation 18 produces

$$\mathbf{J}\mathbf{J}^T \lambda = \dot{\mathbf{p}}_0 - \mathbf{J}\dot{\boldsymbol{\nu}}_0, \quad (20)$$

which we can solve for the Lagrange multipliers λ . These, in turn, can be used to obtain the camera differential for the current time-step

$$\dot{\nu} = \dot{\nu}_0 + \mathbf{J}^T \lambda. \quad (21)$$

Once the derivative is computed, we use Euler's method of integration to solve for the new camera parameters. We write this as

$$\nu(t + \Delta t) = \nu(t) + \Delta t \dot{\nu}(t). \quad (22)$$

In order to make a user selected point follow the mouse, we modify Equation 20 by placing a constraint on $\dot{\mathbf{p}}$ such that

$$\dot{\mathbf{p}} = \dot{\mathbf{p}}_0 - k_f (\mathbf{p}_0 - \mathbf{p}), \quad (23)$$

where \mathbf{p}_0 is the mouse position, \mathbf{p} is the projected point that is being dragged, and k_f is a scale factor. Effectively, this attracts the selected point to the mouse pointer and helps control some of the numerical drift. We use $\dot{\mathbf{p}}_0$ to represent the amount of mouse motion in the current time-step. With this in mind, we can then rewrite the equation as

$$\mathbf{J}\mathbf{J}^T \lambda = \dot{\mathbf{p}}_0 + \dot{\mathbf{p}} - \mathbf{J}\dot{\nu}_0. \quad (24)$$

4.6. Construction of the Jacobian matrix

The Jacobian matrix for our viewing transformation can be written as

$$\mathbf{J} = \begin{pmatrix} h'(\mathbf{V}\mathbf{x}_i) \mathbf{P} \frac{\partial \mathbf{T}}{\partial t_x} \mathbf{Q} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \mathbf{P} \frac{\partial \mathbf{T}}{\partial t_y} \mathbf{Q} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \mathbf{P} \frac{\partial \mathbf{T}}{\partial t_z} \mathbf{Q} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_x} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_y} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_z} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_w} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \frac{\partial \mathbf{P}}{\partial f_x} \mathbf{T} \mathbf{Q} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \frac{\partial \mathbf{P}}{\partial f_y} \mathbf{T} \mathbf{Q} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \frac{\partial \mathbf{P}}{\partial f_z} \mathbf{T} \mathbf{Q} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \frac{\partial \mathbf{P}}{\partial r_0} \mathbf{T} \mathbf{Q} \mathbf{x}_i \\ h'(\mathbf{V}\mathbf{x}_i) \frac{\partial \mathbf{P}}{\partial c_0} \mathbf{T} \mathbf{Q} \mathbf{x}_i \end{pmatrix}^T, \quad (25)$$

where \mathbf{V} , \mathbf{P} , \mathbf{T} , and \mathbf{Q} are the matrices described in 10 and 11 and \mathbf{x}_i is a control point on the model. There is one such entry for each point being controlled (either dragged or locked) by the user. The methods for constructing the derivative matrices are shown in [4, 11].

For moded operation, we simply set the derivatives which are not associated with the selected mode to zero. Thus, for translation, rotation, and scaling modes we get

$$\mathbf{J}_t = \begin{pmatrix} h'(\mathbf{V}\mathbf{x}) \mathbf{P} \frac{\partial \mathbf{T}}{\partial t_x} \mathbf{Q} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \mathbf{P} \frac{\partial \mathbf{T}}{\partial t_y} \mathbf{Q} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \mathbf{P} \frac{\partial \mathbf{T}}{\partial t_z} \mathbf{Q} \mathbf{x} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T, \quad \mathbf{J}_q = \begin{pmatrix} 0 \\ 0 \\ 0 \\ h'(\mathbf{V}\mathbf{x}) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_x} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_y} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_z} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \mathbf{P} \mathbf{T} \frac{\partial \mathbf{Q}}{\partial q_w} \mathbf{x} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T, \quad (26)$$

and

$$\mathbf{J}_p = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ h'(\mathbf{V}\mathbf{x}) \frac{\partial \mathbf{P}}{\partial f_x} \mathbf{T} \mathbf{Q} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \frac{\partial \mathbf{P}}{\partial f_y} \mathbf{T} \mathbf{Q} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \frac{\partial \mathbf{P}}{\partial f_z} \mathbf{T} \mathbf{Q} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \frac{\partial \mathbf{P}}{\partial r_0} \mathbf{T} \mathbf{Q} \mathbf{x} \\ h'(\mathbf{V}\mathbf{x}) \frac{\partial \mathbf{P}}{\partial c_0} \mathbf{T} \mathbf{Q} \mathbf{x} \end{pmatrix}^T, \quad (27)$$

respectively.

4.7. Calibration for stereo displays

The calibration of the stereo display system is a straightforward extension of this approach. The stereo system consists of a pair of cameras which have a parallax due to their different poses (i.e., positions and orientations). To calibrate the stereo display system, we use the above calibration procedure to estimate the parameters for the left and right displays independently. This will account for the different rigid transformations for the poses of the two cameras that represent the two eyes as well as for the differences in the focal lengths for the left and right eyes. The final scene is displayed using the resulting camera parameters estimated using this process.

5. Experimental verification for calibration

A serious problem with the verification of an optical see-through display calibration is that it is not possible to show how well the model corresponds with the object for a human viewer. Since we can't read the image from the back of our retina, there is no way to quantitatively express the

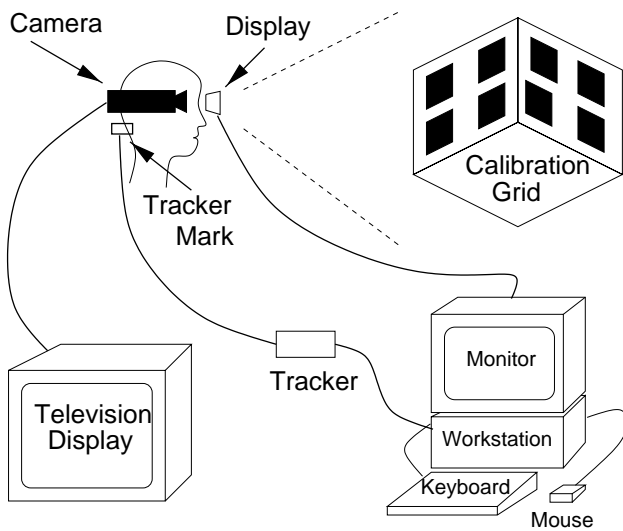


Figure 6. A schematic drawing of our experimental setup showing the various components.

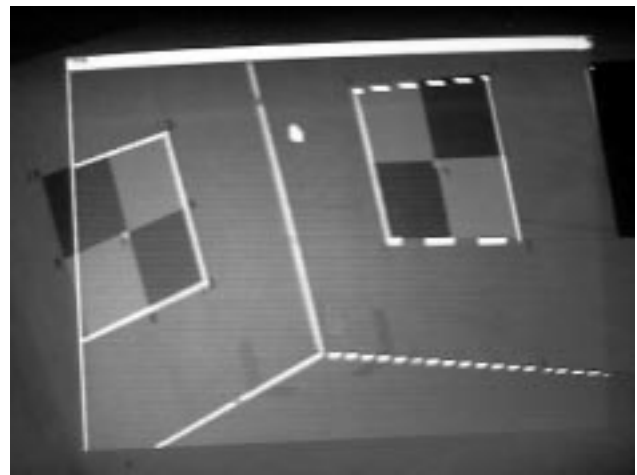


Figure 7. Alignment error image from the viewpoint in which calibration was done (all control points are visible). The alignment errors in this case are minimal.

error. We have thus devised an experiment in which a camera replaces the human eye. In this section, we will give the details of this experiment as well as our results.

5.1. Experimental setup

To set this experiment up, we mounted a camera inside the head of a mannequin where the eye would be located. We then attached the head to a camera tripod and placed the *i-glasses*TM/marker assembly onto it as if it were a real person. The head could be rotated through the use of the tripod controls. Note once again, that this experiment was performed strictly for the purpose of reporting our results and gathering quantitative error measures. In the real setup, the user wears the see-through display/tracker mark assembly and their eye becomes the camera for which we estimate the parameters. Figure 6 shows a schematic of the entire setup.

The graphics were sent to the see-through display so that the camera received both the virtual and real objects simultaneously. This is similar to what the user would see during the calibration procedure. The camera signal was sent to a monitor, which gave the user feedback from his adjustments. Figures 7, 8, and 9 show grabbed frames from the camera.

The user adjusted the virtual camera by dragging the model control points until they align with the corresponding points on the alignment grid.

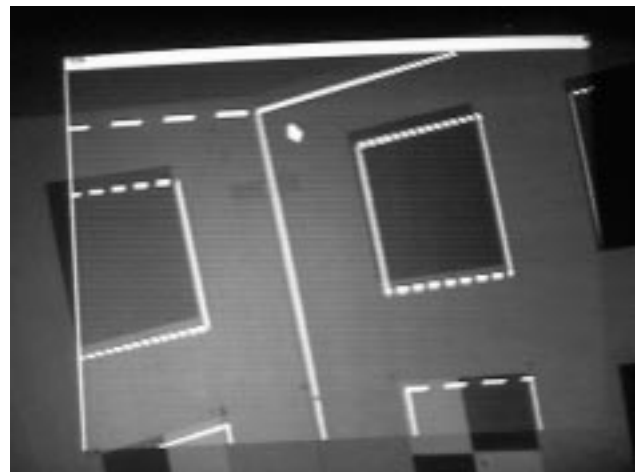


Figure 8. Alignment error image for target square with center (0,8,18) (in inches). This view is obtained by rotating the camera straight up from the viewpoint in Figure 7.

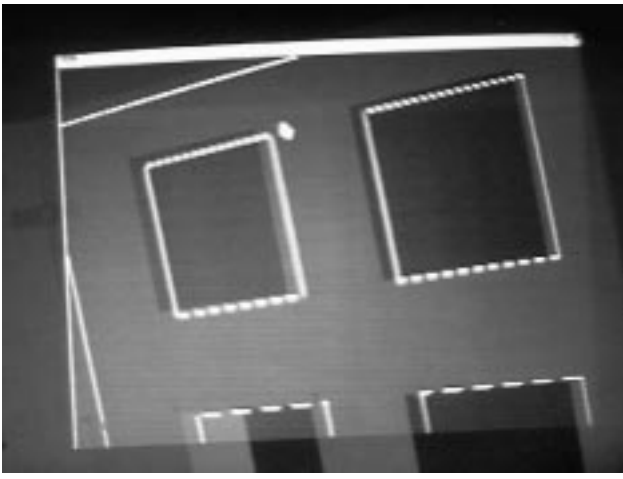


Figure 9. Alignment error image for target square with center (0,18,18) (in inches). This view is obtained by rotating the camera up and to the right from the viewpoint in Figure 7.

5.2. Results

A calibration consists of a sequence of steps. First, the camera is placed in some arbitrary location facing toward the calibration grid so that all control points are visible. Then the control points of the model are dragged until they are aligned with the projected images of their corresponding points on the grid. This dragging causes the solver to perform estimates of the translation, rotation and scaling parameters for the virtual camera. Once all the points are aligned, the estimated parameters of the virtual camera should match those of the real camera and the calibration sequence is complete.

Figure 7 shows an example calibration result using our method. We see in this figure the image of the geometric model of the calibration object superposed on the image of the physical calibration object using the estimated camera parameters. Figures 8, and 9 show the same calibration object model from different viewpoints by moving the dummy head/camera/*i-glasses*TM assembly. As can be seen in these results, the perfect alignment of the calibration object in one view is destroyed in the other views. This is partly due to our particular experimental setup. The calibration done from a single viewpoint does not yield sufficient information to get all the scaling ambiguities correctly. When the calibration is performed dynamically on a human head, however, where the head is free to move and look at the calibration object from multiple views as the interactive calibration process proceeds, the calibration results improve considerably.

In addition to these qualitative results shown in the Fig-

ures 7–9, we ran a set of experiments to collect quantitative error information. In this procedure, we calibrated the camera independently ten times, each time starting from a fresh initial setup. After each calibration was complete, we captured sample images from three different viewpoints of the calibration model projected and superposed on the image of the calibration object. One viewpoint was the one in which the calibration was performed. The second viewpoint was obtained by moving the camera straight up from the calibration viewpoint. The third one was obtained by moving the camera up and to the right from the calibration viewpoint. Figures 7–9 are representative examples of these viewpoints. We then collected a number of prominent points, such as the corners of the squares, in each sample image (10 images per viewpoint) and computed the amount of error in pixels between the projected virtual point using the calibration results and the images of the corresponding points on the physical calibration object.

Table 1 shows the mean and variance of the distances between the selected virtual points and real points for each of the three views. A pixel with this configuration on the physical object corresponds roughly to 1/25th of an inch.

Viewpoint	δ (Pixels)	σ^2
1	2.85	10.06
2	11.44	8.60
3	21.57	31.81

Table 1. The alignment error statistics computed from the data collected using our experimental setup. The table shows the mean and variance of the alignment error over ten independent experiments for each view. The error measure is the Euclidean distance in pixels between the image location of the projected landmark model points and their physical counterparts.

6. Conclusion

In this paper, we presented a camera calibration procedure for optical see-through head-mounted displays for augmented reality systems. Because in augmented reality systems we do not have direct access to the image produced on the retina, the procedure needs to use indirect methods to do the calibration. The method presented in this paper uses a dynamic, interactive framework, in which the user can modify the camera parameters until the image of a calibration object model is aligned with the image seen by the human eye of the physical calibration object.

The resulting calibrations using this method are acceptable within the calibration volume, but the errors increase as the camera moves outside the calibration volume. The quality of the calibrations seem to be better when done on a human head as they are intended, instead of the artificial setting we have for the purposes of collecting quantitative data.

Currently, the interaction is done in a moded fashion (i.e., translation only, rotation only, scaling only, etc). A better way would be to have the user interact in a more free manner by just dragging the control points to where they should be and let the system solve for the camera parameters. We have implemented this approach, but we have found that this results in very jittery and sometimes unstable and counterintuitive motions of the object in the image. The user seems to have a hard time controlling the objects in this case. We did implement the weighted version of the through-the-lens control method described by Gleicher in [4]. This seemed to improve the interaction quite a bit, but still did not quite solve all the difficulties. We could not find a good, non ad hoc way of determining the weights.

We are currently working on a modification of the moded interaction which we hope will improve and streamline the calibration process. We also expect the accuracy will be improved.

7. Acknowledgments

We thank the European Computer-Industry Research Centre (ECRC) for allowing us to use their Grasp system in our research.

References

- [1] R. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.
- [2] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through display. *Computer Graphics*, pages 194–204, July 1994.
- [3] T. Caudell and D. Mizell. Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 659–669, 1992.
- [4] M. Gleicher and A. Witkin. Through-the-lens camera control. *Computer Graphics*, pages 331–340, July 1992.
- [5] S. Gottschalk and J. Hughes. Autocalibration for virtual environments tracking hardware. *Computer Graphics*, pages 65–72, August 1993.
- [6] R. Holloway. *An Analysis of Registration Errors in a See-Through Head-Mounted Display System for Craniofacial Surgery Planning*. PhD thesis, University of North Carolina at Chapel Hill, 1994.
- [7] A. Janin, D. Mizell, and T. Caudell. Calibration of head-mounted displays for augmented reality applications. In *Proc. of the Virtual Reality Annual International Symposium (VRAIS'93)*, pages 246–255, 1993.
- [8] K. N. Kutulakos and J. R. Vallino. Affine object representations for calibration-free augmented reality. In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 1996.
- [9] R. K. Lenz and R. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-10:713–720, 1988.
- [10] P. G. Maillot. Using quaternions for coding 3d transformations. In *Graphics Gems*, pages 498–515, 1990.
- [11] E. McGarrity and M. Tuceryan. A method for calibrating see-through head-mounted displays for augmented reality. Technical Report TR-CIS-0499-13, Indiana University Purdue University Indianapolis, April 1999.
- [12] M. Tuceryan, D. Greer, R. Whitaker, D. Breen, C. Crampton, E. Rose, and K. Ahlers. Calibration requirements and procedures for a monitor-based augmented reality system. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):255–273, September 1995.
- [13] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-14(10):965–980, 1992.