Object Calibration for Augmented Reality

Ross T. Whitaker
Chris Crampton
David E. Breen
Mihran Tuceryan
Eric Rose

ECRC-95-04

Technical report ECRC-95-04
(To be published in Eurographics '95 Proceedings, Maastricht, NL, August 1995)

Object Calibration for Augmented Reality

Ross T. Whitaker
Chris Crampton
David E. Breen
Mihran Tuceryan
Eric Rose

European Computer-Industry Research Centre GmbH (Forschungszentrum)
Arabellastrasse 17
D-81925 Munich
Germany
Tel. +49 89 9 26 99-0
Fax. +49 89 9 26 99-170
Tlx. 52 69 10

For more information please contact : Ross T. Whitaker
ross@ecrc.de

Abstract

Augmented reality involves the use of models and their associated renderings to supplement information in a real scene. In order for this information to be relevant or meaningful, the models must be positioned and displayed in such a way that they align with their corresponding real objects. For practical reasons this alignment cannot be known a priori, and cannot be hard-wired into a system. Instead a simple, reliable alignment or calibration process is performed so that computer models can be accurately registered with their real-life counterparts. We describe the design and implementation of such a process and we show how it can be used to create convincing interactions between real and virtual objects.

1 Introduction

Augmented reality (AR) is a technology in which a user's view (or vision) of the *real* world is enhanced or augmented with additional information generated from a computer model. In contrast to virtual reality, augmented reality brings the computer into the "world" of the user rather than immersing the user in the world of the computer.

Augmented reality is dynamic; as the "view" of the user or camera changes, the computer-generated image does likewise. This computer-generated image is not simply a graphical overlay but is a representation of data that is closely tied to the real-life scene and rendered in such a way that the relationship between the real and virtual entities is made apparent. Indeed, an important goal of AR is that the real-world objects and virtual entities should blend together in a manner that appears natural to the user. In many AR applications, this representation takes the form of a photo-realistic rendering of virtual entities, modeled using attribute values that correspond to the "reality" of the user. The success of this illusion depends on a number of factors including:

- the quality of the computer graphics,
- the modeling of the geometry and the optics of the video camera (or see-through display),
- the modeling of the scene's lighting and other environmental attributes, and
- the modeling of the geometry of the real-life scene, including the *alignment* or *registration* of that geometry with the scene it represents.

We consider the latter of these factors to be crucial to the success of an AR system, especially if we want to incorporate interaction between the real and virtual objects. Such interactions add greatly to the realism of the users' experience and are a focus of on-going research [6].

An augmented reality system can be complex, consisting of a variety of computer graphics, computer vision, and video components. The locations and internal parameters for each component are not guaranteed to remain constant from one application to another. There are a number of unknown variables that must be determined, including the non-uniform scaling of the scan converter, the intrinsic parameters of the camera, the dimensions of pointing devices, and the location of tracked and stationary objects.

In this paper, we address the problem of registering or calibrating a geometric model to a single rigid real-world object. We use *object calibration* to mean the calculation of the transformation between an object, as represented by a geometric model, and some other known coordinate system, which we

generally call $world$ coordinates. We assume that object calibrations are performed infrequently, most likely at application set-up time. Once a calibration is complete, the registration of a moving object can be maintained with a real-time tracking technology, such as those relying upon optics, ultrasound, or magnetics. These tracking technologies are likely to require the initial positioning or determination of the relative position of the object with respect to some kind of mark, perhaps a magnetic receiver or a set of light-emitting diodes.

Such an alignment might be obtained, for instance, by fixing a tracker rigidly to an object and making precise physical measurements of its location and orientation. However, this approach is impractical for two reasons. First, measurements of sufficiently high precision are difficult and time-consuming to obtain. While such an approach might suffice for a one-time demonstration, it does not provide a general, reusable solution. The second problem of physical measurements is that many tracking devices, such as the Ascension "Flock of Birds" system we are using, do not have a well-defined intrinsic coordinate system that can be located on the hardware. For example, precisely where in the 50cm $\times$ 50cm $\times$ 50cm transmitter is the origin of the tracking system? Indeed, we have found that the position of the origin and the orientation of the axis vary when changing the parameters of the tracking equipment. This problem is unique to augmented reality and is not comprehensively addressed in the work on VR. To deal with these calibration issues, we have constructed a system that is flexible and reusable, allowing different models to be interchanged and calibrated on the fly. Our methods also provide mechanisms for positioning static objects, such as furniture or buildings, that do not lend themselves to physical measurement or conventional tracking technologies.

In order to produce a general object calibration capability, we have developed two separate techniques for determining the location of objects in a real environment and aligning geometric models to them. The first technique utilizes algorithms originally developed for computer vision and involves manually picking known landmarks in images obtained from a video camera. The second technique utilizes a $3D$ pointing device to locate the position of the landmarks in world coordinates. In both approaches the calibration procedure consists of establishing the rigid transformation that maps the local coordinates of a set of landmarks points into the associated measured world coordinate system. Noise in the input data and the landmarks is an important issue that complicates the computation of a final result.

This paper proceeds by reviewing some of the previous work in augmented reality which highlights the importance of calibration. We then describe the Grasp system, an augmented reality system being developed at ECRC, and review some of the calibrations that make this system effective. We then focus on the mathematics and the methodology for calibrating geometric models and rigid objects and show how these calibrations can give rise to a convincing blend of graphics and real-world images.

2 Previous Work

A number of researchers have investigated specific augmented reality applications. Feiner et al. [9] have used augmented reality in a laser printer maintenance task. In their example, the augmented reality system aids the user in the steps required to open the printer and replace various parts. Wellner [20] has demonstrated an augmented reality system for office work in the form of a virtual desk top on a physical desk. He interacts on a physical desk both with real and virtual documents. Bajura et al. [5] have used augmented reality in medical applications in which the ultrasound imagery of a patient is superimposed on the patient's video image. Lorensen et al. [15] use an augmented reality system in surgical planning applications. Drasic et al. [8] use augmented reality with computer-generated stereo graphics to perform telerobotics tasks. Each of these applications involves overlaying graphics on video images and interacting with real-world objects. These applications highlight the need for accurate and robust methods for registering geometric models to objects in a real environment.

Calibration has been an important aspect of research in augmented reality, as well as in other fields, including robotics and computer vision. Deering [7] has explored the methods required to produce accurate high resolution head-tracked stereo display in order to achieve sub-centimeter virtual to physical registration. Azuma and Bishop [4], and Janin et al. [14] describe techniques for calibrating a see-through head-mounted display. Gottschalk and Hughes [11] present a method for auto-calibrating tracking equipment used in AR and VR applications. Gleicher and Witkin [10] state that their through-the-lens controls may be used to register $3D$ models with objects in images. Grimson et al. [13] have explored vision techniques to automate the process of registering medical data to a patient's head.

3 The Grasp System

The Grasp system has been developed at ECRC as a platform for research in augmented reality [1, 2]. It has been used to develop several prototype AR applications, including a mechanic's assistant [18] and a collaborative tool for interior design [3]. The system provides functionalities in the areas of $3D$ graphics, image processing, six degrees-of-freedom (in 3-space) tracking, and real-time $3D$ interaction.

A schematic of the Grasp hardware configuration is shown in Figure 3.1. The graphical image is generated by the workstation hardware and displayed on the workstation's high resolution monitor along with auxiliary control information, i.e., buttons and panels. A scan converter takes the relevant portion of the graphical image and converts it to a standard video resolution and format. The scan converter also mixes this generated video signal with the video input from

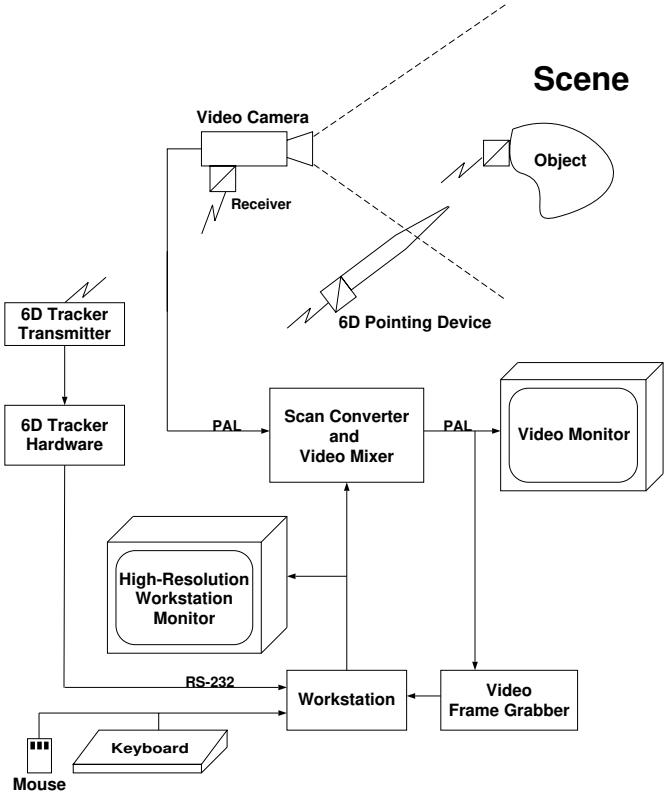


Figure 3.1: The Grasp system hardware configuration.

the camera. A $3D$ 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 including the video camera, a pointing device, and other objects of interest. A frame grabber is used to grab digital video images for processing within the system. The Grasp system currently uses Sun workstations with "Flock of Birds" magnetic trackers from Ascension Technologies, an Otto scan converter from Folsom Research, and Sun VideoPix frame grabber hardware. The software has been implemented using the C++ programming language.

For an AR system such as Grasp to be effective it must include complete and accurate calibration methodologies. For example, objects rendered by the computer using a virtual camera whose intrinsic parameters do not match those of the real camera will result in images that are not realistic when viewed in the context of an AR application. Likewise, the mis-registration of a model with the object it represents can cause artifacts that undermine the intended visual effect.

3.1 Required Transformations

A list of the various coordinate systems in a typical Grasp application helps us to understand the calibration steps that are required to establish a complete mapping between the real and virtual worlds.

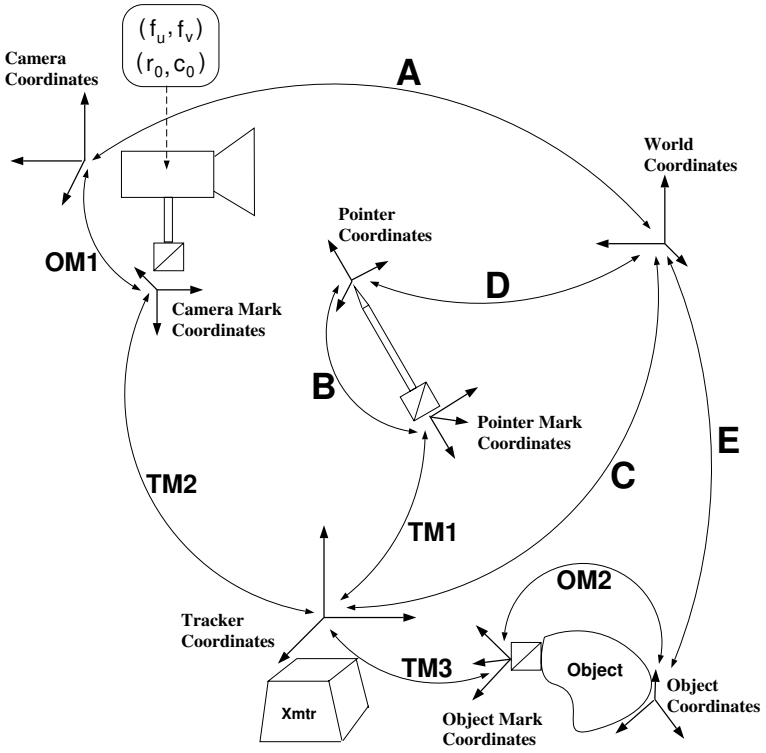


Figure 3.2: Grasp coordinate systems and their relationships.

Transform	Description	Persistence	Calculation Process
A	world-to-camera	varying	landmark point correspondence
B	pointer-mark-to-pointer	fixed	orientation measurement
C	world-to-tracker	fixed	from B and TM1
D	world-to-pointer	varying	from B, C and TM1
E	world-to-object	varying	landmark point correspondence
TMx	tracker-to-mark	varying	from tracker system
OMx	object-to-mark	fixed	from C, E (or A), and TMx

Table 3.1: Key transforms between Grasp coordinate systems.

Figure 3.2 shows each of the main coordinate systems used by Grasp and their relationship to each other. Table 3.1 lists the important transforms between these coordinate systems, their persistence over time, and how they are originally obtained. The central reference is the $world$ coordinate system which is at a fixed and known location relative to the operating environment.

The camera has an associated coordinate system which determines the formation of images. The location of this coordinate system within the world coordinate system must be calculated in order for the virtual or digital camera to produce renderings of virtual objects that are consistent with the incoming video images. The world-to-camera transform relating the two coordinate systems is labeled "A" in the diagram.

The reference coordinate system used by the $3D$ tracker is referred to as the

"Tracker" coordinate system and remains at a fixed location while the system is operating. Its relationship to world coordinates is defined by the world-to-tracker transform which is labeled "C" in Figure 3.2. Tracked objects have a receiver (or mark) rigidly attached to them, and the tracking system generates data describing the tracker-to-mark transforms, represented by the arcs labeled "TM1", "TM2" and "TM3" (TMx) in the diagram.

This paper focuses on the computation needed to determine "E". In the case of a moving object being tracked, "E" is combined with tracker readings at the time of calibration to yield "OM2", which is constant over time as long as the tracker is rigidly fastened to the object.

For each real object that interacts with entities in the virtual world, we need a model to act as its virtual counterpart. Each model is defined within its own coordinate system. In order to register a virtual object with its real-world counterpart, we need to know the world-to-object transform for that object. Therefore, during the execution of applications, transforms "A", "D" and "E" are of primary interest. However, the application is able to measure only transforms "TMx" directly (from the tracker). Updated values for "A", "D" and "E" can be computed by combining the tracker data with the other transforms that have been calculated during the calibration procedures such as "B" and "OMx". The object to world transforms are calculated during the object calibration process. For tracking real objects we calculate the fixed, object-to-mark (OMx) transform by combining the world-to-object with tracker readings (TMx) taken at the same time. Once we have the object-to-mark transform, we can continually update the object-to-world transform as TMx varies. Note that in some applications not all world-to-object transforms are varying because in some applications real objects may be "static" within the scene.

The pointer object is a particular case of a tracked object with its coordinate system centered on the pointer tip, and transform "B" representing the transform between the coordinate system of the pointer and that of the tracker mark. Because the pointer is used as part of the process of locating the tracker transmitter within the world coordinate system, the procedure for calibrating the pointer is somewhat different from the procedures for other tracked objects. It relies on several orientation measurements around a fixed point[2].

3.2 Calibration Procedures

The following is a sequence of calibration processes for calculating each of the transforms discussed in the previous section:

1. image calibration,
2. camera calibration (transform A),
3. pointer calibration (transform B),

4. tracker transmitter calibration (transforms D and C),

5. object calibration (transform E),

6. tracker mark calibration, one per tracked entity (all transforms OMx).

The *image calibration* determines the $2D$ image distortion introduced into the system by the scan converter and frame grabber. Subsequent calibration procedures that use "grabbed" images from the camera require a model of this distortion. More detailed descriptions of the techniques used for this and the other calibration procedures that are not given in this paper are described elsewhere[2][19].

The *camera calibration* process determines the camera location and orientation as well as the intrinsic camera parameters (focal length, image center and aspect ratio). This process calculates the transform labeled "A" in Figure 3.2.

Pointer calibration calculates the geometry of $3D$ pointing device used within an application. In our current system, this pointer object is a wand with a tracking mark (receiver) attached at the base. In particular, this calibration step calculates the position of the tip of the pointer relative to the tracker mark (the transform labeled "B"). This calibration is important because several other calibration steps rely on the $3D$ pointing device.

Tracker transmitter calibration calculates the position and orientation of the tracker's coordinate system within the world coordinate system (the transform represented by the arc labeled "C" in Figure 3.2). This transform is calculated indirectly after calculating the transforms "D" and "B" and measuring transform "TM1".

The *mark calibration* calculates the transform between the coordinate system of the tracker mark and the coordinate system of an object (transforms "OMx") using the measured transform "TMx", along with the previously calculated world-to-object transforms ("E", for example). This transform is fixed for as long as the object and mark are rigidly attached to one another. Non-tracked real-world objects are expected to remain static so their virtual counterparts can function as "scenery" in the virtual world.

Object calibration is the process whereby the location and orientation of a real-world object is calculated such that a virtual counterpart can be placed at the corresponding location, with the appropriate orientation, within the virtual world (transform "E" in the diagram). Some real-world objects will subsequently have their movements tracked by their virtual "shadows", in which case the corresponding tracker marks must also be calibrated.

4 Object Calibration

Object calibration is the procedure whereby the virtual entities within the computer's world are *registered* with their real-world counterparts. There are two particular cases where this is necessary:

- to support "direct" interactions between the user and real-world objects via a computer model (counterpart) of the real object, and
- to support interactions between real and virtual objects that use computer models of the real objects.

An example of the former would be a mechanic querying the status of a component by pointing at the real object and clicking. The application must map the mechanic's action into a logical "pick" on the virtual data in order that the real component can be correctly identified and the user provided with the data requested. An example of the latter would be the occlusion of virtual objects by real-world objects. For instance, if a virtual chair goes behind a real table, the table should occlude appropriate parts of the chair. To support the above functionalities, we first need a computer model of the object and then a calibration procedure to locate the real object so that the virtual model can be registered to it.

The calibration procedures described here require a number of *landmark points* – points whose positions are known in the coordinate system of the object model. Geometric models of objects might be created piece-by-piece from a set of geometric primitives or they might come from a CAD system. Regardless of their origin, models are generally stored as files on disk for subsequent use. Therefore files describing real objects must contain, in addition to the geometric and attribute data, the $3D$ positions and labels of the landmark points. These points should correspond to features on the object, such as corners or creases, that can be easily identified by a user. The registration procedure consists of locating the corresponding points on the real object and the model, and then calculating the object-to-world transformation from these point pairs.

4.1 Image-based Object Calibration

The goal of an image-based object registration is to start with a calibrated camera and compute the object-to-camera transformation of a single object for which there is a known geometric model. The position of an object is determined "through the lens" of the camera. The camera can be the same camera that is used to capture the video signal that is combined with the graphics. The calibration begins by capturing an image of the real-world object and locating a set of landmark points in the image.

There has been extensive research in pose determination in the computer vision literature [12, 16], but most of these techniques apply to only limited classes of models and scenes. The focus of the computer vision research is typically automation and recognition, concepts that are interesting, but not essential to AR. In our work, the locations of landmark points in the image are found manually by a user with a mouse. We assume that the points are mapped from known locations in 3-space to the image via a rigid $3D$ transformation and a projection.

4.1.1 Camera Model

We model the camera with a pinhole camera model shown in Figure 4.1. This model defines the basic projective imaging geometry with which the object landmarks are mapped onto the $2D$ image plane. This model is a simplification of the optics of a real camera and is used often in computer graphics and computer vision to describe the formation of images in a camera. There are different ways of setting up the coordinate systems; we use a right-handed coordinate system, where 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 similar triangles:

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

The camera model used for the generation of the graphical images must be related to the pixel coordinate system of the display device. The relationships between the screen coordinates, the pinhole model, and the world coordinates (shown in Figure 4.2) must be clearly defined in order to establish the precise relationship between a pixel location in an image and the $3D$ ray which projects to that point.

The position and orientation of the camera with respect to the world coordinate system (O, x, y, z) is defined by a rigid transformation consisting of a rotation $\mathbf{R}$ and a translation $\vec{T}$:

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \mathbf{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \vec{T}. \tag{2}$$

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

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

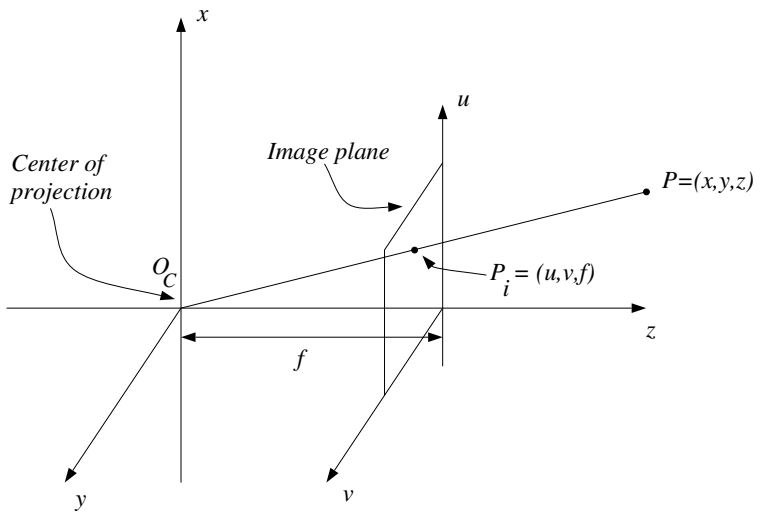


Figure 4.1: The geometry of the simple pinhole camera model for perspective transformation.

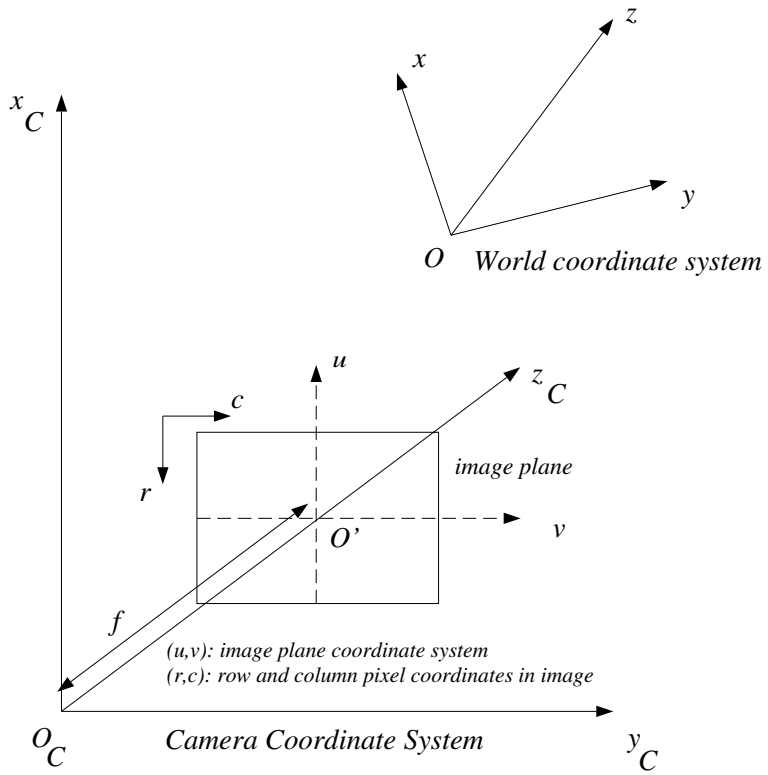


Figure 4.2: The various imaging coordinate systems with respect to each other.

where (r_0, c_0) are the pixel coordinates of the image plane origin O'. The terms s_u and s_v serve two purposes. First, they account for the proper sign (note that the r-axis and u-axis are pointing in opposite directions) of the coordinate axis. Second, they allow for the non-isotropic nature of some cameras thereby capturing the aspect ratio. 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 T are known as the *extrinsic camera parameters*.

4.1.2 Numerical Solutions

The goal of our work is to determine the position of the object, which contains a set of landmark points, in some known coordinate system. Because the camera is assumed to be calibrated, its coordinate system is known. Therefore, finding the position of the object in camera coordinates is identical to the camera calibration problem (which has been studied extensively in the literature [17]), except that we assume the internal parameters, f_u, f_v, r_0, and c_0, are known. The result is a linear system which follows from Equations 2 and 3,

$$\begin{aligned}(r_i - r_0)x_i r_{31} + (r_i - r_0)y_i r_{32} + (r_i - r_0)z_i r_{33} + (r_i - r_0)t_3 & \\ -f_u x_i r_{11} - f_u y_i r_{12} - f_u z_i r_{13} - f_u t_1 &= 0 \quad \text{and} \quad (4)\\ (c_i - c_0)x_i r_{31} + (c_i - c_0)y_i r_{32} + (c_i - c_0)z_i r_{33} + (c_i - c_0)t_3 & \\ -f_v x_i r_{21} - f_v y_i r_{22} - f_v z_i r_{23} - f_v t_2 &= 0, \quad (5)\end{aligned}$$

where $(r_i, c_i, x_i, y_i, z_i)$ are the pixel and $3D$ coordinates of the landmark points.

There are numerous methods for solving this system. The simplest is to divide by t_3 and treat it as a non-homogeneous linear system and solving for the 11 unknowns, r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}, t_1, and t_2, to within a scale factor t_3. We can assume $t_3 > 0$ (the object is in front of the camera), and use the fact $\mathbf{R}$ is a rotation matrix (i.e., $r_{31}^2 + r_{32}^2 + r_{33}^2 = 1$) to find t_3 after we have solved for the other eleven parameters. The linear system can be over-constrained to account for error by choosing more than six landmarks and finding a least-squares solution.

We have found, in practice, that this approach does not work. The errors introduced by the mouse clicks in the image, the measurements of the model, and the camera calibration give solutions that are not rigid transformations, even with as many as 15 points. A subsequent fix of $\mathbf{R}$ that forces it to be rigid results in excessive error.

Thus, Equations 4 and 5 must be solved while accounting for the fact that $\mathbf{R}$ is a rotation matrix. One approach is to express $\mathbf{R}$ in terms of three degrees of freedom which span the space of rotation matrices (Euler angles for instance). However, this produces a nonlinear system with singularities that make finding a solution difficult. Instead we treat Equations 4 and 5 as a homogeneous linear system with 12 unknowns and simultaneously enforce rigidity by solving

the nonlinear optimization problem

$$\min_{\vec{x}} ||\mathbf{A}\vec{x}||^2 + \alpha||\mathbf{R}\mathbf{R}^T - \mathbf{I}||^2, \tag{6}$$

where $\vec{x}$ is the set of twelve unknowns in the rigid transformation ($\mathbf{R}$ and $\vec{T}$), and α is a term which controls the amount of rigidity. The $2n \times 12$ matrix $\mathbf{A}$ comes directly from equations 4 and 5, and n is the number of landmark points.

In practice, the precise value of α ($\alpha = 1$ is chosen for our work) has very little effect on the solutions. With the solution from the unconstrained linear system as an initial condition, either a gradient descent or Newton-Raphson algorithm solves this optimization problem in a reasonable amount of time (fractions of a second). We have found this method produces rigid transformations (to within 0.1% which can be corrected afterwards without noticeable side effects) and point matches that are within 2-3 pixels when solutions are re-projected onto the image plane.

Despite good pointwise alignment in the image plane, the image-based calibration can produce significant error in the t_3 term which is not seen in the re-projected solutions. For instance, in the case of the engine model shown in Figure 4.3(a), the image-based approach can produce a rigid transformation which matches landmark points in the image to within about 2 pixels. Yet the error in the z-direction (distance from the camera) can be as much as 2-3 centimeters. This error becomes evident as the object is turned as in Figure 4.3(b). We attribute this error primarily to error in the camera calibration, and better camera models and calibration procedures are a topic of ongoing research. Because of this problem, we have developed the procedure described in the next section for calibrating objects using a $3D$ pointing device.

4.2 Pointer-Based Object Registration

For this method we assume a pointing device which is capable of generating the world coordinates of $3D$ positions in the real world. In our case, the pointing device (which itself must already be calibrated) is a magnetic tracker attached to a wooden cone which, when calibrated, has an accuracy of about $\pm$1cm. Over half of this error is bias and transfers directly into object calibration error.

The problem here is to compute the rigid transformation between a set of $3D$ point pairs. Using the $3D$ pointer and several keystrokes the user indicates the world coordinates (or some other $known$ coordinate system), P_i^w, of landmark points on the object, which are given in the object's local coordinate system as P_i^l. The relationship between these sets of points,

$$P_i^w = P_i^l\mathbf{R} + \vec{T} \tag{7}$$

gives rise to a linear system of 12 unknowns ($\vec{x}$). For a unique solution 4 points

are needed, but in most cases we use more than 4 points and solve for the least-squares error.

As with the image-based object calibration, error in the measurements can produce solutions that represent non-rigid transformations. Such non-rigidities can produce undesirable artifacts when this transformation is combined with others in the graphics system. As in the case of image-based calibration, we force $\mathbf{R}$ to be a rotation matrix by solving the following non-linear optimization problem:

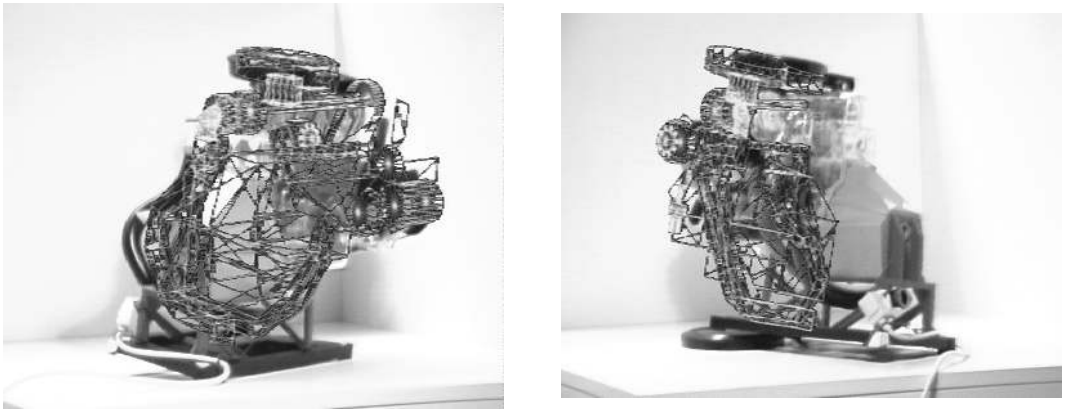
$$\min_{\vec{x}} ||P_i^w - P_i^l\mathbf{R} - \vec{T}||^2 + \alpha||\mathbf{R}\mathbf{R}^T - \mathbf{I}||^2. \quad (8)$$

This procedure provides the object-to-world transformation that is needed for object registration within a small number of seconds. Figure 4.4(a) shows a model engine which has been calibrated in such a manner. Rotations of the engine (Figure 4.4(b)) show that this calibration does not suffer from the $depth\ problem$ of the image-based approach.

5 Results

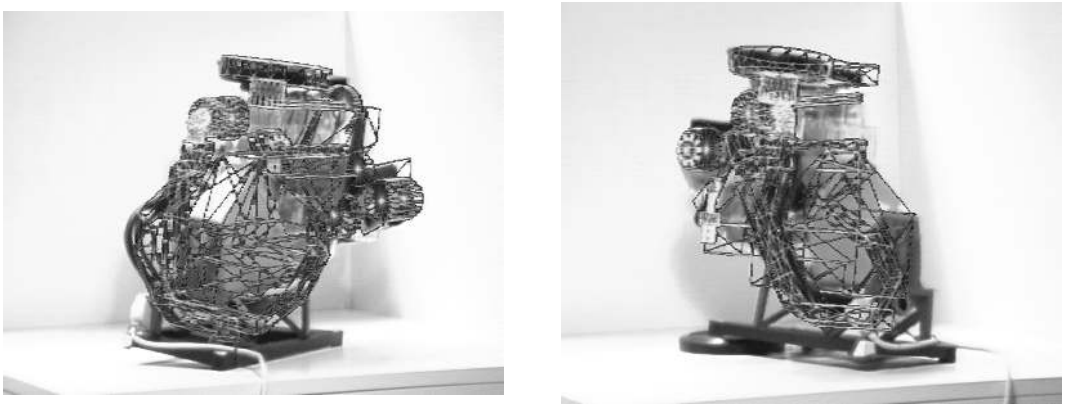
The calibration procedures outlined above have been implemented as an integral part of the Grasp system. Several applications have been developed that utilize the object registration procedures, including a mechanics' assistant [18] and an interior design tool [3]. Figure 5.1 presents the results of a successful object calibration. A geometric model of an automobile engine has been registered to a real model engine using a calibrated pointer. Correct registration allows us to properly label the visible components of the real engine. As the engine is rotated, visibility calculations are performed on the underlying model, informing the system which components are visible. Only the visible components are subsequently labeled.

When dealing with large physical objects, such as the furniture used in the interior design application, the image-based calibration proves more practical. Figure 5.2(a) shows a wireframe model of a table overlaid on the image of a real table after registration. In this case, image-based registration has been performed by picking known calibration points in an image grabbed by a calibrated camera. The landmark points, which are the corners and legs of the table, have been highlighted with circles to show that they were chosen. Figure 5.2(b) shows the same real table occluding two virtual chairs as an example of augmented reality with interaction between real and virtual objects [6]. This effect is obtained by properly aligning the $3D$ model of the table with the physical table as shown in Figure 5.2(a) and using the graphics system to perform the occlusion shown in Figure 5.2(b).



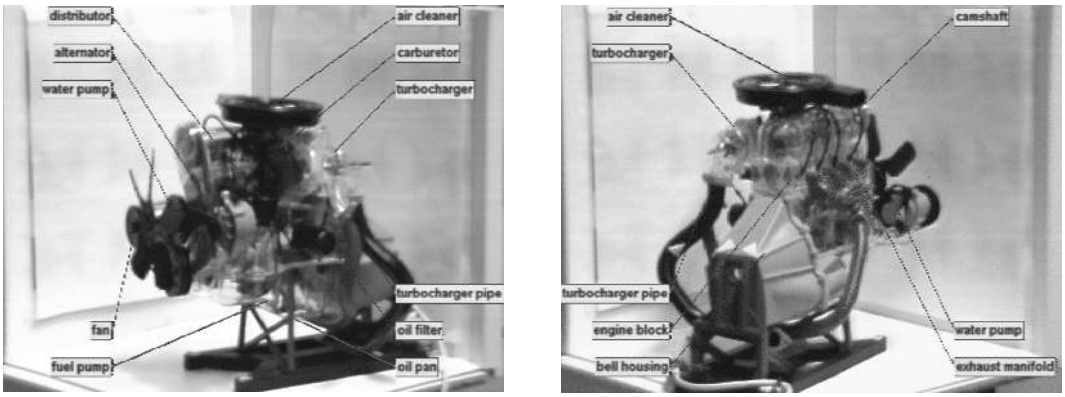
(a) (b)

Figure 4.3: Calibration and tracking of an engine model: A wireframe engine model registered to a real model engine using an image-based calibration (a), but when the model is turned and its movements tracked (b), the graphics show the misalignment in the camera's z-direction.



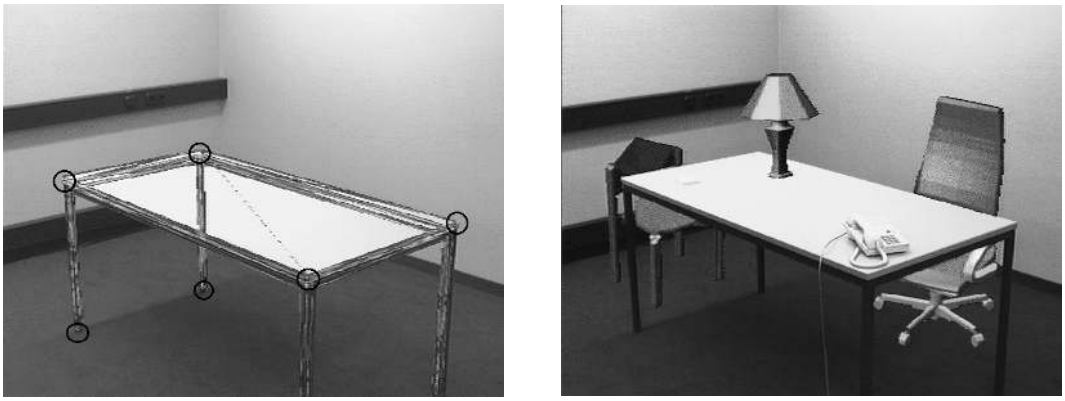
(a) (b)

Figure 4.4: (a) A wireframe engine model registered to a real model engine using an pointer-based calibration. (b) When the model is turned and its movements tracked, the graphics show the correct alignment in the camera's z-direction.



(a) (b)

Figure 5.1: (a) A real model engine annotated in augmented reality and (b) As the model is turned only visible components are annotated.



(a) (b)

Figure 5.2: (a) A wireframe table model registered to a real table and (b) A real table occluding two virtual chairs.

6 Conclusion

Augmented reality is a powerful technology that provides new paradigms for human-computer interaction and communication. The calibration of devices and objects is an essential component of augmented reality. Without reliable and accurate calibration procedures, convincing augmented reality effects are not possible. In this paper we have focused on one of these calibration procedures: object calibration. Here, we calculate the transformation that maps a geometric model, in a local coordinate system, into the world coordinate system such that the model and the real-world object are precisely registered. With this registration, after blending the computer-generated graphical images with the live video signal, we can produce effects such as occlusion and collision detection.

We have presented the procedures and mathematics for two object calibration techniques. The first technique utilizes computer vision algorithms to determine the pose of the object. A user manually identifies known landmarks in a digital image of the object that has been grabbed through a calibrated camera. These point correspondences are used to calculate the rigid $3D$ transformation that aligns the object and the model. We have found that this technique suffers from excessive error in the direction perpendicular to the focal plane of the camera. In the second technique, a calibrated pointing device is used to locate the object landmarks in the world coordinate space. Correlating the two sets of $3D$ points produces accurate transformations. In both techniques noise in the input data is a problem, and we have demonstrated methods for coping with it. An accurate calibration can be used to produce a variety of powerful visual effects. Our future work will focus on automating and improving the calibration procedures. This will involve automatically locating landmarks in images, and improving the accuracy and stability of our numerical procedures.

7 Acknowledgments

The AutoCAD model of the automobile engine was initially produced by Anne Bachy. The plastic automobile engine model was generously provided by Revell, Inc. This work is financially supported by Bull SA, ICL PLC, and Siemens AG.

Bibliography

[1] K. Ahlers, D. Breen, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer. An Augmented Vision System for Industrial Applications. In *SPIE Photonics for Industrial Applications Conference Proceedings*, October 1994.

[2] K. Ahlers, C. Crampton, D. Greer, E. Rose, and M. Tuceryan. Augmented Vision: A Technical Introduction to the Grasp 1.2 System. Technical Report ECRC-94-14, ECRC, Munich, Germany, 1994.

[3] K. Ahlers, A. Kramer, D. Breen, P.-Y. Chevalier, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer. Distributed Augmented Reality for Collaborative Design Applications. Technical Report ECRC-95-03, ECRC, Munich, Germany, 1995. To appear in: Proceedings of Eurographics '95, Maastricht, NL.

[4] R. Azuma and G. Bishop. Improving Static and Dynamic Registration in an Optical See-through Display. In *Computer Graphics (Proc. SIGGRAPH)*, pages 194–204, July 1994.

[5] M. Bajura, H. Fuchs, and R. Ohbuchi. Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery within the Patient. In *Computer Graphics (Proc. SIGGRAPH)*, pages 203–210, Chicago, IL, July 1992.

[6] D. Breen, E. Rose, and R. Whitaker. Interactive Occlusion and Collision of Real and Virtual Objects in Augmented Reality. Technical Report ECRC-95-02, ECRC, Munich, Germany, 1995.

[7] M. Deering. High Resolution Virtual Reality. *Computer Graphics (Proc. SIGGRAPH)*, 26(2):195–202, July 1992.

[8] D. Drascic, J. J. Grodski, P. Milgram, K. Ruffo, P. Wong, and S. Zhai. Argos: A Display System for Augmenting Reality. In *Formal Video Programme and Proceedings of Conference on Human Factors in Computing Systems (INTERCHI'93)*, page 521, Amsterdam, Netherlands, 1993.

[9] S. Feiner, B. MacIntyre, and D. Seligmann. Knowledge-Based Augmented Reality. *Communications of the ACM*, 36(7):53–62, July 1993.

[10] M. Gleicher and A. Witkin. Through-the-Lens Camera Control. In *Computer Graphics (Proc. SIGGRAPH)*, pages 331–340, Chicago, IL, July 1992.

[11] S. Gottschalk and J. Hughes. Autocalibration for Virtual Environments Tracking Hardware. In *Computer Graphics (Proc. SIGGRAPH)*, pages 65–72, August 1993.

[12] W.E. Grimson. *Object Recognition by Computer*. The MIT Press, Cambridge, MA, 1990.

[13] W.E. Grimson, T. Lozano-Perez, W. M. Wells, G. J. Ettinger, S. J. White, and R. Kikinis. An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 430–436, Seattle, WA, June 1994.

[14] A. Janin, D. Mizell, and T. Caudell. Calibration of Head-Mounted Displays for Augmented Reality Applications. In *Proceedings of IEEE VRAIS '93*, pages 246–255, September 1993.

[15] W. Lorensen, H. Cline, C. Nafis, R. Kikinis, D. Altobelli, and L. Gleason. Enhancing Reality in the Operating Room. In *Visualization '93 Conference Proceedings*, pages 410–415, Los Alamitos, CA, October 1993. IEEE Computer Society Press.

[16] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, 1985.

[17] S. J. Maybank and O. D. Faugeras. A Theory of Self Calibration of a Moving Camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

[18] E. Rose, D. Breen, K.H. Ahlers, C. Crampton, M. Tuceryan, R. Whitaker, and D. Greer. Annotating Real-World Objects using Augmented Vision. Technical Report ECRC-94-41, ECRC, Munich, Germany, 1994. To appear in: Proceedings of Computer Graphics International '95, Leeds, UK.

[19] M. Tuceryan, D. Greer, R. Whitaker, D. Breen, C. Crampton, E. Rose, and K. Ahlers. Calibration Requirements and Procedures for Augmented Reality. Technical Report ECRC-95-14, ECRC, Munich, Germany, 1995.

[20] P. Wellner. Interacting with Paper on the Digital Desk. *Communications of the ACM*, 36(7):87–96, July 1993.