2002

Computer vision-based registration for augmented reality.

Zonglei. Huang
University of Windsor

Computer Vision-based Registration for Augmented Reality

ZONGLEI HUANG

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2002

Abstract

Augmented reality is a young but very active research area. It is considered a variant of virtual reality. In virtual reality, the user is immersed completely in the virtual scene; whereas in an augmented reality system, the user can see the real scene with the virtual object superimposed onto the real scene.

Registration in augmented reality aligns the virtual object with the real scene in 3D. The use of computer vision techniques for registration is an ideal method to accomplish this. In this thesis, a computer vision-based interactive object registration system was developed for augmented reality. In this system, the user can register 3D virtual object onto real images by reconstructing the registration plane. The registration process is semi automatic and only needs intuitive mouse clicking. Experiments show that the system is easy to use and has very good practical registration accuracy.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction to Augmented Reality

1.1 Definition

Augmented reality (AR) is a variation of virtual reality (VR). It is a young but very active research area. In virtual reality, the user is completely immersed into a virtual scene. In such a system, the user is in a virtual environment and is isolated from reality. While in an augmented reality system, the user can see and interact with the real world as well as with virtual objects superimposed onto the real scene. Therefore, augmented reality systems provide extra information to the real scene presented to the user.

Milgram [67] produced a taxonomy showing how work in VR and AR are related; he describes it as the reality-virtuality continuum as shown in figure 1.1.

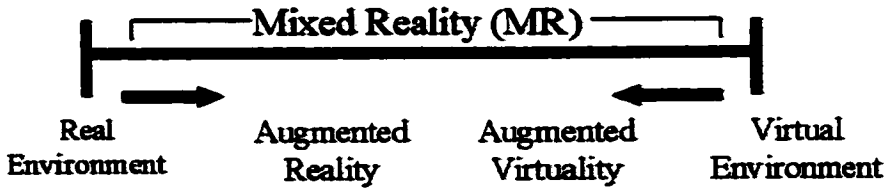

Figure 1.1: Milgram's reality-virtuality continuum

Figure 1.2 is the screen snapshot of a Java3D program written to demonstrate the

basic idea of superimposing a virtual object onto a real scene. The user can change the position and orientation of the virtual cube by rotation and translation operations. The cube, as a result, appears naturally on the table.

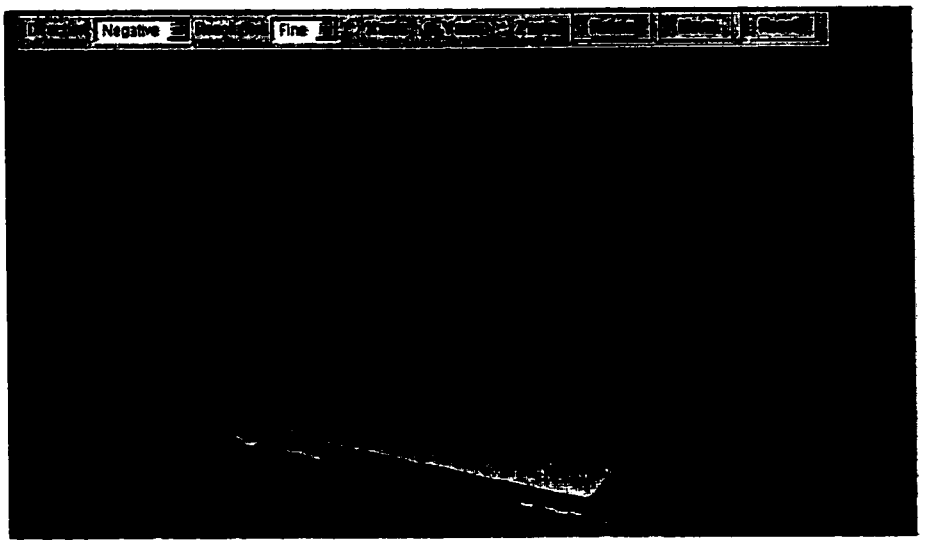

Figure 1.2: Superimposing virtual object(a cube) onto real scene

In the AR community, we generally accept the following characteristics for augmented reality system [4]:

- Combines real and virtual
- Interactive in real-time
- Registered in 3D

1.2 Different Types of AR System

An augmented reality system combines the virtual and real scene together and presents the augmented scene to the user. Different combining and display technologies make different types of augmented reality systems, for example, optics-based augmented reality system and video-based augmented reality system.

1.2.1 Optics-based AR System

A see-through head-mounted display(HMD) system is a typical optics-based augmented reality system. It looks like a helmet with the optical combiner in front of the user's eyes. The optical combiner plays two roles: one is transmitting the light from the real world to the user (so that the user can see the real scene directly), and the other is reflecting the virtual scene to the user. Thus, the real and virtual scenes are combined at the optical combiner. The design of the optical combiner is complicated, since it is both partially transmissive and reflective. Figure 1.3 shows the conceptual diagram of an optical see-through HMD augmented reality system.

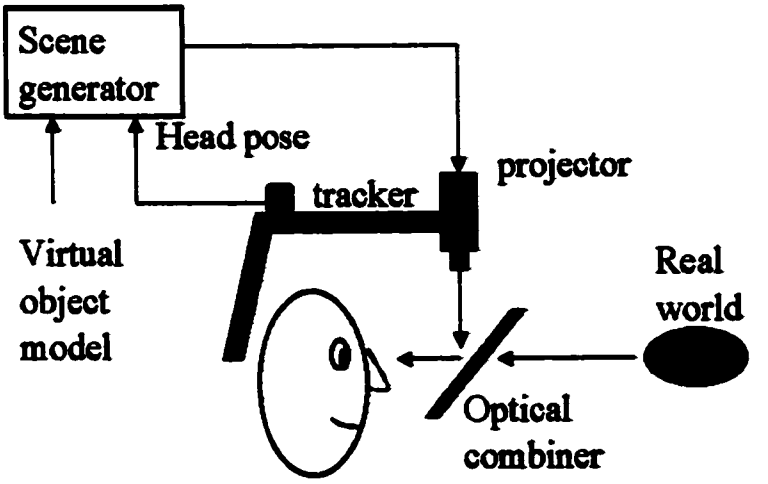

Figure 1.3: Optical see-through HMD augmented reality system

1.2.2 Video-based AR System

Instead of combining scene optically, a video-based AR system performs it by video composition. For example, a video see-through HMD system uses one or two head-mounted video cameras to provide the user's view. Before the video stream is displayed in the monitor, it is blended with the images of virtual objects that are generated by a computer. The augmentation of virtual objects on real scene is thus achieved.

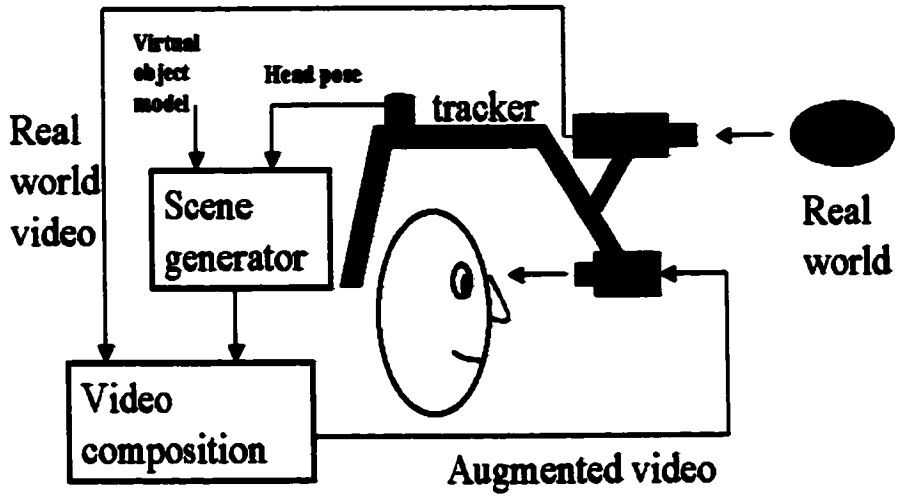

Figure 1.4: Video see-through HMD augmented reality system

In the video-based AR system, the use of cameras and video composition systems makes it flexible for composition and registration strategies. The camera usually provides wider field-of-view than the combiner used in the optics-based AR system. Furthermore, because the real scene reaches the user indirectly, the delay of the blending will not be too sensitive to the user and is easier to be compensated for.

On the other hand, the optics-based AR system does not use camera. This has some advantages: first, the system is simplified and the display resolution will not be degraded by the video camera system. Second, the user will not lose his sight even though the system malfunctions. The user can still see the real scene directly through the combiner. Hence, optics-based AR systems are safer than the video-based AR systems.

1.3 Registration Problem

The term registration in AR means to align the virtual object properly with the real environment in 3D. Virtual objects must behave as their real counterparts would behave. For example, the cube on the table in figure 1.2 must be situated in and stay in

the proper place as the user moves around in the environment and it should disappear when another object (real or virtual) obstructs the user's view.

The registration process can be done in three steps: positioning, rendering and merging. Positioning is to put the virtual object at a desired position, and rendering is to get the 2D image of the positioned 3D virtual object. The rendering of the virtual object will then be merged with the image of real environment to compose an augmented image.

Before we register a virtual object on a desired position, we have to know the coordinates of that location. However, the measurement of the coordinates is usually not a straightforward process. First of all, the metric 3D information of the real environment is not readily available. Second, the measurement needs high accuracy. In a video-based augmented reality system, the image of the real environment is captured by a camera, which is considered as a 3D to 2D projection of the real environment. Similarly, the 3D virtual object will be rendered by a virtual camera that simulates the real camera. Thus, the augmentation is achieved by a merging of the two projections. Knowing the projection matrix of the real camera is a necessary step in the registration process.

Registration errors will directly lead to a misalignment. Over all, these errors can be roughly grouped into two categories: static errors and dynamic errors. Static errors are the errors that have effects even the user's viewpoint or the scene is completely still; while dynamic errors are the errors that have effects when the user's viewpoint or the scene is moving. Among those sources of errors, the following are due to static errors:

- Optical distortion
- Tracking errors
- Projection errors

The optical distortion is caused by the lens of the camera or the display system. Mainly, it is a radial distortion. Radial distortion is due to lenses where the magnification at the edge is different from the one at the center of the lens. Because of lens

shapes, radial distortion is also a function of the lens's field of view. Radial distortion is quite prevalent in wide-angle lenses. The effect is that, for example, a straight line will appear curved when it locates far away from the optical center.

Optical distortion can be computationally compensated for, however, the computation will even lead to a bigger delay-induced registration error than the distortion it corrects [44].

In order to know the position of the user's viewpoint or the moving object in an augmented reality system, a tracking system is used. The tracking system, however, inevitably has errors. That is, the difference between the output of the tracking system and the actual position of the tracking target will cause a misalignment in the registration.

Incorrect projection parameters will also lead to a misalignment in registration. As mentioned above, the virtual object is rendered by a simulated camera whose projection parameters must be exactly the same as the real camera. However, estimating the parameters of the real camera is an error prone process.

Dynamic errors are mainly caused by some kinds of system delay, for example, the delay in the tracking system and rendering system. The heavy computation load and response time of the physical components will cause the system delay. In [44], Richard L. Holloway states that system delay causes more registration errors than all other sources. Therefore, suppressing the system delays is a critical research topic in augmented reality.

In summary, achieving alignment and depicting occlusions have been the primary and most challenging problems in augmented reality. Most proposed solutions had either a limited success or a limited domain of application. This thesis will focus on the registration problem for a video-based augmented reality system.

1.4 Applications of Augmented Reality

Augmented reality has been successfully applied in medical, industry, military and entertainment fields.

In the medical field, doctors use augmented reality to aid surgery. Through some imaging equipment, such as computed tomography (CT) scans or magnetic resonance imaging(MRI), the view of the target internal anatomy is scanned. When a doctor performs a surgery, these images are rendered and combined in real-time with a view of the patient. This, in effect, gives a doctor "X-ray vision" inside a patient.

In industry, augmented reality is used to help assembly, maintenance and repair of complex machinery. Technical manuals are translated into 3D drawing and annotations that can be superimposed in real-time with views of the actual machine. Those augmented views are presented to the worker through a HMD system. When the worker is working, he is being shown step-by-step the tasks that need to be done and how to do them. Some successful systems are the laser printer maintenance application [30], and Boeing's wire bundle assembly application [16].

Augmented reality is also used in tele-operation of a robot. Instead of operating real robot directly, we can first plan its motion by a virtual robot. The results are directly displayed in the real world. Once it is tested, the plan is then applied to the real remote robot.

In entertainment, augmented reality is used to merge a real actor into a 3D virtual environment. In this way, we no long have to build new physical sets. The entire environment can be virtually simulated, and the production cost is greatly reduced. In [62], the authors described a system that makes intelligent virtual creatures who respond to actions.

Advanced technologies are always applied in military first. Actually, the head-up displays (HUD) and helmet-mounted sights (HMS) have been used in military aircraft for many years. In such a system, extra information is superimposed on the real world view for a pilot; for example, the aim parameters are highlighted and stick on different moving targets. In [51], Simon Julier et al. described a battlefield augmented reality system (BARS). The system consists of a wearable computer, a wireless network system and a tracked see-through HMD. The user's perception of the environment is enhanced by superimposing graphics onto the user's field of view. The graphics are registered (aligned) with the actual environment. For example, an augmented view of a building could include a wire frame plan of its interior, icons to represent reported locations of

snipers and the names of adjacent streets.

Numerous successful applications are being reported every year. The application domain of augmented reality is expanding quickly, which in turn stimulates the research of augmented reality.

1.5 Thesis Overview

This thesis is organized as follows. Chapter 2 provides the theoretical background of computer vision techniques for augmented reality. Chapter 3 presents a review of existing registration techniques for augmented reality. Chapter 4 describes our interactive virtual object registration system for augmented reality. Chapter 5 gives the implementation details and experiments results. Chapter 6 is the conclusion and discussion for the future research direction.

Chapter 2

Background for Vision-based AR

2.1 Computer Vision for Augmented Reality

A registration process needs 3D metric information for positioning and rendering. Unfortunately, this information is quite difficult to obtain and track. For example, in some early developed AR system, researchers used magnetic sensors to track and report the camera position and orientation. However, the magnetic sensor is nonlinear in large measurement range and sensitive to metals and magnetic fields. In fact, the biggest obstacle to build effective AR systems is the lack of accurate, long range sensors and trackers that report the locations of the user and the surrounding objects in the environment.

On the other hand, computer vision techniques have the potential to provide the accurate registration data needed by AR systems. By using computer vision techniques, we can easily extract 3D metric information from 2D images; we can build 3D model for the real environment to deal with occlusion; we can also perform dynamic correction for registration by analyzing the augmented images. Overall, computer vision techniques enable us to get 3D information directly from 2D images which is vital for AR systems. In addition, because computer vision extracts information from the images, the output of augmented reality, it is possible to form a close-loop control, which makes the system more stable and accurate.

2.2 Digital Images

In the 1950's, scientists began to develop intelligent machines. The goal of intelligent machines is to simulate natural biological system, which inevitably requires the machine to have vision capability. Computer vision addresses the problem of programming computers to see. It is the discipline that gives interpretations and useful decisions based on digital images taken by cameras. In [95], the scope of the computer vision is defined as: *a set of computational techniques aimed at estimating or making explicit the geometric and dynamic properties of the 3D world from digital images.*

Computer vision uses digital images as inputs. A digital image of an object is created by sensing the light rays reflected from the object's surface. In a CCD(Charged Coupled Device) camera, there is an $n \times m$ array of CCD photo-sensors. When the light ray hits the photo-sensor, it generates different voltage proportional to the intensity of the light. These voltage signals are digitalized and stored in an $n \times m$ integer array. The digital image is thus presented by the $n \times m$ integer array and each entry of the array usually has the value of grey-level (light intensity) from 0 to 255: 0 means black and 255 means white.

2.3 Camera Model and Calibration

The camera projects 3D space points to 2D image points. The optical distortion and digitalization process of a real CCD camera make the modelling far from being perfect. There are quite a few camera models, but the most popular model is the pinhole one. The pinhole model describes a pure perspective projection: 3D space points are projected onto the image plane of the camera, as shown in figure 2.1.

In this model, a reference frame is setup at the camera's optical center, O. The image plane Π is placed in front of the optical center. The optical axis is the line through the point O and perpendicular to the image plane, which is the Z axis of the reference frame. The focal length f is the distance between the optical center O and the image plane. The space point P is projected onto the image plane as point p which is at the intersection point of the image plane and line $< OP >$. The pixel point $p(x, y)$

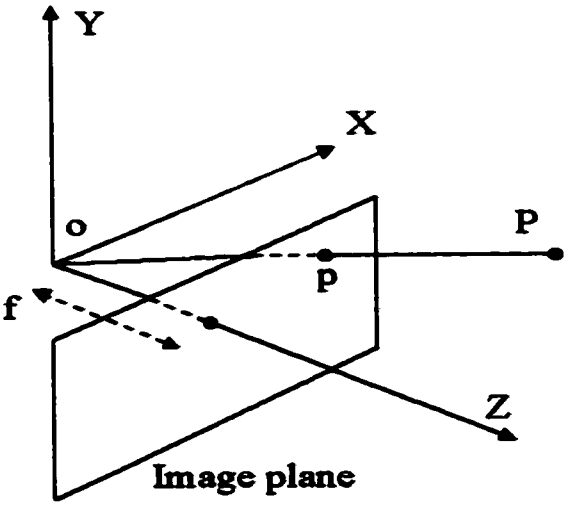

Figure 2.1: The pinhole camera model

and space point $P(X, Y, Z)$ has the following relation:

$$\begin{aligned} x &= f\frac{X}{Z}, \\ y &= f\frac{Y}{Z} \end{aligned} \tag{2.1}$$

In addition to the camera reference frame, there are other reference frames. Their relation is described in figure 2.2. The camera reference frame is located with respect to the world reference frame and the coordinates of the image points in the camera reference frame can be defined in the pixel coordinates.

In computer vision, a camera has two sets of parameters: extrinsic parameters and intrinsic parameters. The extrinsic parameters are a set of geometric parameters that describes the camera's position and orientation with respect to the world reference frame. The extrinsic parameters can be defined as a transformation from the world reference frame to the camera reference frame, which consists of a pure rotation R and a translation T [95]:

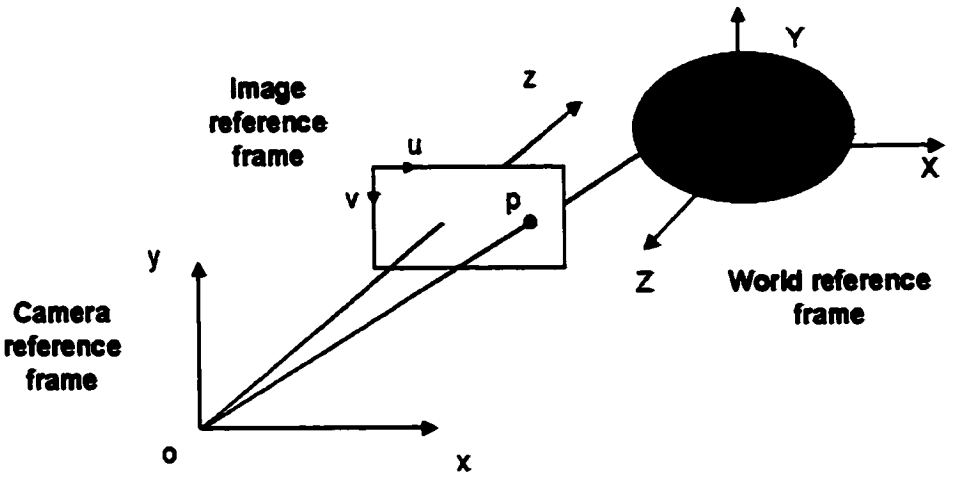

Figure 2.2: The relationship between reference frames

$$P_c = R(P_w - T) \tag{2.2}$$

where P_c and P_w are coordinates of the space point P in the camera reference frame and the world reference frame respectively. R is a 3×3 rotation matrix and T is translation vector:

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix},$$
$$T = \begin{pmatrix} t_{11} \\ t_{21} \\ t_{31} \end{pmatrix} \tag{2.3}$$

Intrinsic parameters describe the optical, geometric and digital characteristics of the camera. For a pinhole camera, they are [95]:

1. Focal length f
2. Transformation from camera reference frame to pixel coordinates
3. Geometric distortion introduced by the optics

Neglecting the distortions, the pixel coordinates (x_{img}, y_{img}) are related to their camera reference frame coordinates (x, y) by:

$$\begin{aligned} x &= -(x_{img} - o_x)s_x, \\ y &= -(y_{img} - o_y)s_y \end{aligned} \tag{2.4}$$

where (o_x, o_y) is the image center in pixel units, s_x and s_y is the horizontal and vertical direction scale factor. In summary, there are 5 intrinsic parameters: f, o_x, o_y, s_x, s_y.

Camera calibration is a fundamental problem in computer vision and is a necessary step to extract metric information from 2D images. It is a process of determining the intrinsic and extrinsic parameters of the camera, or the process of estimating the projection matrix.

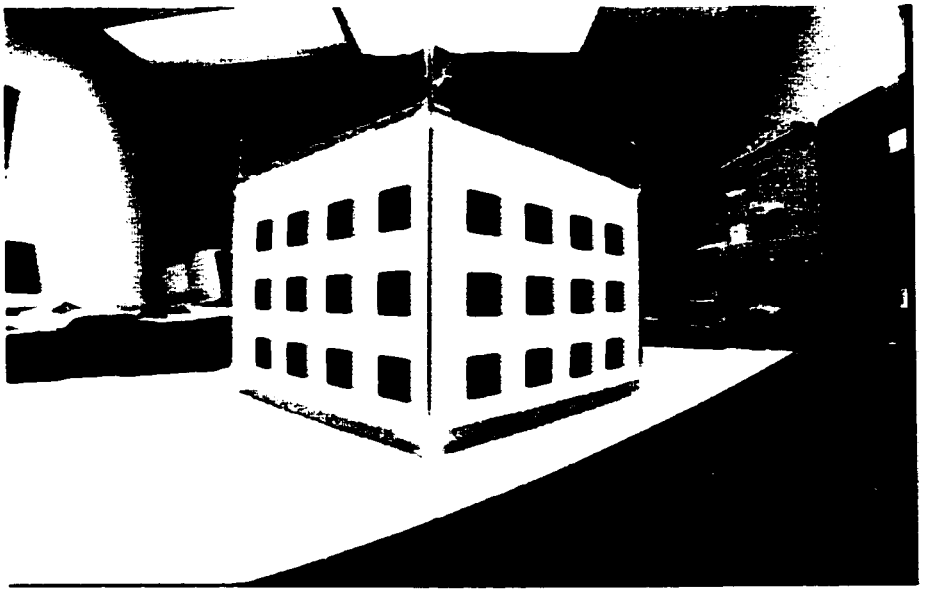

Figure 2.3: The calibration pattern

Camera calibration techniques are roughly classified into two categories: pattern-based calibration and self-calibration [105]. Pattern-based calibration uses a special calibration object, called calibration pattern. The calibration pattern's geometry in 3D space is known; for example, in figure 2.3, the 3D coordinates of the feature points (the corners of the black squares on the pattern) are known. Once a digital image of the calibration pattern is taken by the camera, the correspondences between the 3D feature points and their 2D projections will be built. This set of data contains projection information and will be used to derive the intrinsic and extrinsic parameters. Pattern-based camera calibration can be done very efficiently. A well known example is Tsai's calibration algorithm [96]. A recently improvement by Zhang's calibration algorithm in [105] makes it possible to use a planar calibration pattern, which considerably improves the flexibility.

Self-calibration does not use the calibration pattern. The calibration process is to move the camera rigidly and take images with fixed intrinsic parameters; the correspondences between three images are sufficient to recover both intrinsic and extrinsic parameters. This approach is quite flexible, but it is not mature yet. Because there are many parameters to estimate, we cannot always obtain reliable results [105].

A perspective projection matrix is the link between the 3D space points and the 2D pixel points. A 3D point P must undergo the following steps to be projected onto the image plane:

- A Euclidian transformation from world reference frame to camera reference frame. It is expressed as matrix D, in which 6 extrinsic parameters are held.
- A 3D-2D perspective projection, projecting point P onto the image plane within the camera reference frame. It is expressed as matrix I.
- A 2D-2D transformation from camera reference frame to image coordinates. It is expressed as matrix A, in which 4 intrinsic parameters are held.

Therefore, the projection matrix M can be expressed as a 3×4 matrix M:

$$M = AID \tag{2.5}$$

The projection of a space point $P = (X, Y, Z, 1)$ onto the image point $p = (u, v, 1)$ can be expressed as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \lambda AID \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.6}$$

where λ is the scale factor. Equation (2.6) provides two basic projection equations describing the relationship between points defined in space (X, Y, Z) and their corresponding pixels (u, v):

$$\begin{aligned} u &= \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\ v &= \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \end{aligned} \tag{2.7}$$

where m_{ij}, $i = 1...3$, $j = 1...4$ are the entries of projection matrix. The projection matrix has 9 entries, but only 8 are independent. Therefore, at least 6 points are needed to solve the equation. A singular value decomposition(SVD) is a good method to solve the equation:

$$\begin{pmatrix} -X_1 & -Y_1 & -Z_1 & -1 & 0 & 0 & 0 & 0 & u_1X_1 & u_1Y_1 & u_1Z_1 & u_1 \\ 0 & 0 & 0 & 0 & -X_1 & -Y_1 & -Z_1 & -1 & v_1X_1 & v_1Y_1 & v_1Z_1 & v_1 \\ -X_2 & -Y_2 & -Z_2 & -1 & 0 & 0 & 0 & 0 & u_2X_2 & u_2Y_2 & u_2Z_2 & u_2 \\ 0 & 0 & 0 & 0 & -X_2 & -Y_2 & -Z_2 & -1 & v_2X_2 & v_2Y_2 & v_2Z_2 & v_2 \\ & & & & & & \vdots & & & & & \\ -X_n & -Y_n & -Z_n & -1 & 0 & 0 & 0 & 0 & u_nX_n & u_nY_n & u_nZ_n & u_n \\ 0 & 0 & 0 & 0 & -X_n & -Y_n & -Z_n & -1 & v_nX_n & v_nY_n & v_nZ_n & v_n \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix} = 0 \tag{2.8}$$

It is possible to re-cast the problem of solving equation 2.8 as a nonlinear minimization problem, where we attempt to minimize the distance in the image plane between the points (u_i, v_i) and the re-projected points (x_i, y_i). This can be done by defining the merit function:

$$E = \sum_{i=1}^{N} \|x_i - u_i\|^2 + \|y_i - v_i\|^2 \rightarrow min \tag{2.9}$$

In general, non-linear methods lead to more robust solutions. In our program, the Levenberg-Marquardt nonlinear minimization method is used to solve the equation.

2.4 Epipolar Geometry

After we get the projection matrix through camera calibration, it is easy to calculate the projection of a 3D space point, as described by equation (2.7).

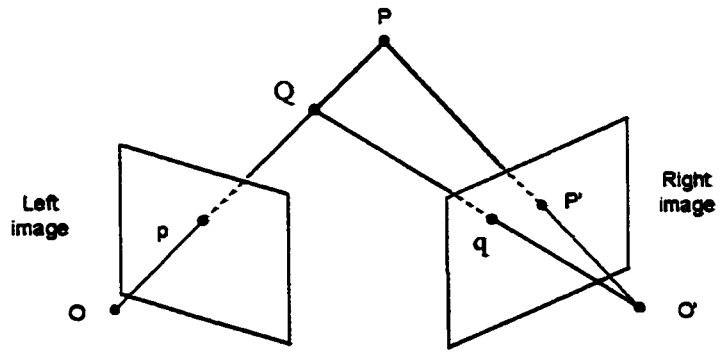

Figure 2.4: Recovering the depth from two views

Figure 2.4 shows that both point P and Q are projected on the image point p, which means that a space point's 3D coordinates can not be recovered from its single 2D projection. However, if we are provided two views, as shown in figure 2.4, the situation is different. We can recover the 3D coordinates of space point p by calculating the intersection of the two lines $< pP >$ and $< p'P >$, where p and p' are the projections of space point P on left image and right image respectively.

Stereo vision is concerned with how to infer 3D information of a scene from two images from different viewpoints. Stereo vision must solve two primary problems:

- Determine which item in the left image corresponds to which item in the right image.

- Reconstruct the 3D coordinates of the inspected points.

Epipolar geometry is a tool to help solving the first problem. Figure 2.5 shows two pinhole camera models. As usual, each associates a 3D reference frame. The space point P has the projection p on the left image plane and p' on the right image plane respectively. The line connects two optical center O and O' has the intersection with two image planes: e and e'. These two image points are called epipoles. Actually e' is the projection of O on the right image plane and e is the projection of O' on the left image. Line $< pe >$ on the left image plane is the projection of line $< PO' >$, and line $< p'e' >$ on the right image plane is the projection of line $< PO >$. These two lines are called epipolar line.

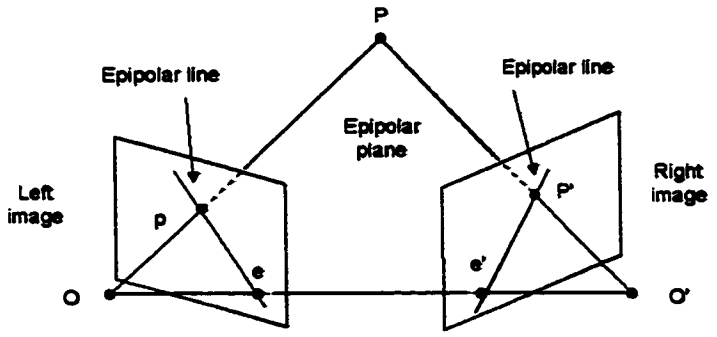

Figure 2.5: Epipolar geometry

From figure 2.5, one fact is clear: O, O', p and p' are coplanar. The cross product of vector $\overrightarrow{OO'}$ by vector $\overrightarrow{O'p'}$ is a vector perpendicular to the plane defined by O, O', and p. Therefore, the following equation holds:

$$\overrightarrow{O'p'} \cdot (\overrightarrow{O'O} \wedge \overrightarrow{Op}) = 0 \tag{2.10}$$

Equation (2.10) is true only if all the vectors are defined in the same reference frame. Here we select the right camera as our reference frame, and denote $\overrightarrow{O'p'}$, $\overrightarrow{O'O}$ and $\overrightarrow{Op}$ by p'^T, t and p. Thus, equation (2.10) becomes equation (2.11):

$$p'^T \cdot ([t]_\times Rp) = p'^T TRp = 0$$
$$T = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{2.11}$$

where R is the 3×3 rotation matrix from left reference frame to the right reference frame, and T is the 3×3 antisymmetric matrix. The above relation is usually written as equation (2.12):

$$p'^T Ep = 0 \tag{2.12}$$

where E is a 3×3 matrix, called essential matrix. The geometry interpretation of the essential matrix gives out the epipolar constraint. Since Ep is a 3×1 vector, $(a, b, c)^T$, and if $p'^T = (x, y, 1)^T$, equation (2.12) becomes $ax+by+c = 0$, which states that point p' on the right image plane belongs to the line defined by Ep. To find p's corresponding point p' on the right image, we do not need to search the whole right image, but only along its epipolar line. Therefore, epipolar constraint reduces the searching area from 2D to 1D (a line), and thus improves the matching efficiency.

If we replace the normalized coordinates $(x, y, 1)$ of p with corresponding pixel coordinates $(u, v, 1)$, the epipolar constraint becomes:

$$p'^T Fp = 0 \tag{2.13}$$

where F is the fundamental matrix. Equation (2.13) is more practical, since the point coordinates can be easily obtained from the left and the right image. The fundamental matrix has 8 independent parameters, therefore at least 8 pairs of point are needed to solve for F. The following is the equation derived from equation (2.13) that is used to solve for F:

$$\begin{pmatrix} u'_1u_1 & u'_1v_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 1 \\ u'_2u_2 & u'_2v_2 & u'_2 & v'_2u_2 & v'_2v_2 & v'_2 & u_2 & v_2 & 1 \\ & & & & \cdot & & & & \\ & & & & \cdot & & & & \\ & & & & \cdot & & & & \\ u'_nu_n & u'_nv_n & u'_n & v'_nu_n & v'_nv_n & v'_n & u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0 \qquad (2.14)$$

where f_{ij}, $i = 1...3$, $j = 1...3$ are the entries of fundamental matrix. (u_i, v_i) and (u'_i, v'_i), $i = 1...n$ are the coordinates of n pairs corresponding points on the left and the right image.

2.5 Similarity Matching

Epipolar constraint narrows the searching range for the identification of point correspondences between a pair of stereo images. There are other constraints that can further narrow the searching range. The order constraint states that stereo projection always preserves the same order of points along the epipolar line. If image point A is on the right side of point B on the epipolar line, the matching point A' of A must on the right side of matching point B' of B. The continuity constraint states that the disparity will vary smoothly on the object surface and an abrupt change will be considered as the object boundary.

However, all of these constraints only narrow the searching range but can not be used for identifying the exact correspondence between two points in the stereo image. Stereo matching is done by a similarity measurement. An intuitive method is to use correlation function to calculate the differences of grey value of the two compared points and their neighboring pixels. Two points that give minimal differences will be considered the matched points. There are a few correlation functions, mainly differing in how to express the difference in the grey value. For example, SSD uses the sum

of squared difference, SAD uses the sum of absolute difference, ZSSD uses zero mean sum of squared difference and ZSAD uses zero mean sum of absolute difference. An outstanding correlation function, ZNCC (Zero Mean Normalized Cross Correlation) is more robust than the others. It is defined as follows:

$$ZNCC((x,y),(x',y')) = \frac{\sum_{v=0}^{vlen}\sum_{u=0}^{ulen}(R(x'+u,y'+v)-\overline{R})\cdot(S(x+u,y+v)-\overline{S})}{\sqrt{\sum_{v=0}^{vlen}\sum_{u=0}^{ulen}(R(x'+u,y'+v)-\overline{R})^2\cdot\sum_{v=0}^{vlen}\sum_{u=0}^{ulen}(S(x'+u,y'+v)-\overline{S})^2}} \quad (2.15)$$

where R and S are the template windows centered at point (x,y) and (x',y') being compared. *ulen* and *vlen* are the size of template windows, and usually they have same value. $R(x'+u,y'+v)$ and $S(x+u,y+v)$ are the grey value of points in template window R and S respectively. $\overline{R}$ and $\overline{S}$ are the average grey value of two template windows. Aschwanden [3] proves that ZNCC is very robust against many kinds of images distortion and noise.

2.6 3D Reconstruction

After having point correspondences identified, the 3D reconstruction from calibrated stereo images is straightforward. Recall that in the perspective projection, a space point and its 2D projection point is linked by the projection matrix:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.16)$$

If we know the projection matrix and the projections of a space point in both the left and the right image, we can easily recover that space point's 3D coordinates (X,Y,Z)

by solving the following equations:

$$\begin{aligned} u &= \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{43}} \\ v &= \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{43}} \\ u' &= \frac{m'_{11}X + m'_{12}Y + m'_{13}Z + m'_{14}}{m'_{31}X + m'_{32}Y + m'_{33}Z + m'_{43}} \\ v' &= \frac{m'_{21}X + m'_{22}Y + m'_{23}Z + m'_{24}}{m'_{31}X + m'_{32}Y + m'_{33}Z + m'_{43}} \end{aligned} \tag{2.17}$$

we have 4 equations and 3 unknowns. The above equation can be rewritten as equation 2.18:

$$\begin{pmatrix} m_{11} - m_{31}u & m_{12} - m_{32}u & m_{13} - m_{33}u \\ m_{21} - m_{31}v & m_{22} - m_{32}v & m_{23} - m_{33}v \\ m'_{11} - m'_{31}u' & m'_{12} - m'_{32}u' & m'_{13} - m'_{33}u' \\ m'_{21} - m'_{31}v' & m'_{22} - m'_{32}v' & m'_{23} - m'_{33}v' \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} m_{34}u - m_{14} \\ m_{34}v - m_{24} \\ m'_{34}u' - m'_{14} \\ m'_{34}v' - m'_{24} \end{pmatrix} \tag{2.18}$$

Equation 2.18 can easily be solved by SVD method. In summary, the coordinates of a space points can be recovered by knowing its projections on at least two calibrated images.

2.7 Conclusion

In this chapter we first addressed the usefulness of computer vision for augmented reality. Then we reviewed some related computer vision techniques that will be used in the augmented reality registration system. The camera calibration is to recover camera's pose and projection parameters from 2D digital images. Epipolar constraint helps narrow the searching area in similarity matching. All of them will be used for 3D reconstruction, which is a primary step in our registration system.

Chapter 3

Existing Registration Methods

A video-based augmented reality system presents to the user a video stream of the real scene with virtual objects superimposed on it. These virtual objects will look as if they really exist in the real scene. In order to achieve such an indistinguishable blending, the 3D virtual objects are first modelled and placed with the real objects in the same reference frame, the world reference frame. Then, the 3D virtual objects are rendered and the rendering is blended with the image stream of the real scene. Rendering a 3D virtual object is analogous to imaging a real object using a camera. Therefore, for a video-based AR system, in order to keep the consistence while blending the images of virtual object and real scene, the virtual object should be rendered by a simulated virtual camera that has the same intrinsic and extrinsic parameters as the real one. The perceived registration between real and virtual objects depends on how accurately the virtual camera models the real one.

Figure 3.1 shows the detailed transformation models for augmented reality. The virtual object is first positioned by the World-to-Object transformation which specifies its position and orientation with respect to the world reference frame. Then, the virtual object is transformed to the camera reference frame by the World-to-Camera transformation. The World-to-Camera transformation is calculated by a camera pose estimation system that reports the real camera's position and orientation with respect to the world reference frame. Finally, the virtual object is projected onto the image plane by the Camera-to-Image transformation which is calculated by a camera calibra-

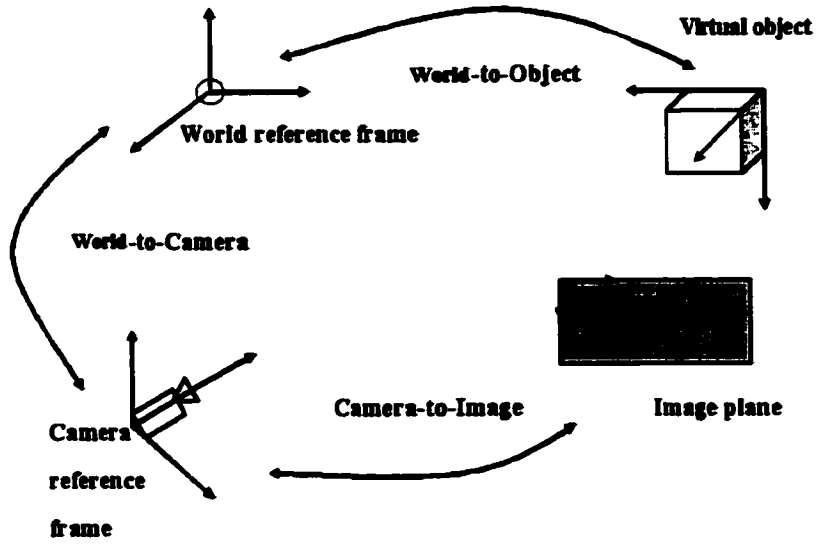

Figure 3.1: Transformation models for augmented reality

tion procedure for the real camera.

In summary, we first establish 3D geometric relationships between the virtual objects and the real world, and then continuously render the virtual objects with their assigned location in 3D space and camera's viewpoint. Thus, the registration process is performed by the following steps:

- Positioning

  This is to know the 3D coordinates where the virtual object is placed. It is a user-computer interactive process and is usually performed at the initialization of the system. The positioning process should be an accurate as well as a user-friendly process. Early researchers were more concerned by how to obtain accurate 3D coordinates of the desired location, for example by directly assigning the 3D coordinates of the location. However, this makes the system defective in user-computer interaction. Recent research has realized those drawbacks, and tries to develop accurate and user-friendly positioning techniques.

- Camera pose estimation

The virtual object's rendering will be merged with the video stream of the real scene to achieve the augmentation. Rendering the virtual object is not trivial work, since it not only depends on its 3D location and orientation, but also depends on the camera's location and orientation. When the camera undergoes a transformation, the parameters for rendering will also be accordingly changed. Therefore, for a real-time augmented reality system, the camera pose must be dynamically tracked at frame rate.

- Virtual camera simulation

  The virtual camera is simulated using the real camera's intrinsic and extrinsic parameters that are obtained through camera calibration and pose estimation procedure.

In this chapter, we will discuss a few registration methods adopted in early and recently developed AR system, in particular focusing on positioning, pose estimation, and virtual camera simulation techniques. In addition, we will also discuss future research trends and the dynamic correction techniques.

3.1 Positioning

3.1.1 Specifying Coordinates

The positioning is to specify the 3D coordinates of the position where the virtual object is going to be registered. There are three reference frames in a video-based augmented reality system: the world reference frame, camera reference frame and the image reference frame. The virtual object will be registered with the real scene in the world reference frame. The simplest and most accurate way for positioning is by directly assigning the position's 3D coordinates with respect to the world reference frame. For example, figure 3.2 shows a system that has the origin of the world reference frame at a corner of the table. To put a cube on the table, one can manifestly specify its 3D coordinates by a metric measurement.

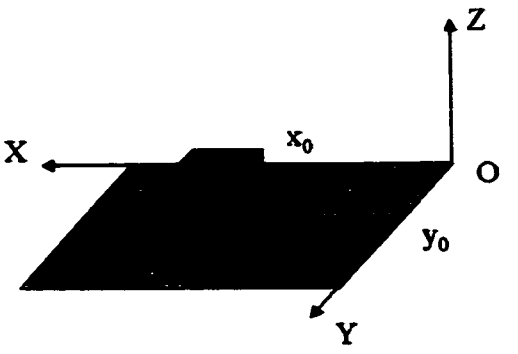

Figure 3.2: Directly specifying the coordinates for the virtual object

However, the measurement is not trivial. First, the origin of the world reference frame may not always be made conveniently for the measurement. Second, the user might not be able to express meaning by using accurate coordinates. For example,"a little bit left" can hardly be converted into a number. Actually, this method will lead to a poor human-computer interaction performance.

3.1.2 Use of Manipulator

The use of the manipulator is an improvement compared to the method of direct measurement. In this method, the user can change the virtual object's position and orientation by a manipulator. Figure 3.3 shows the demonstration developed in Java3D. By clicking the rotation or translation button, the user can rotate or move the virtual cube along a selected axis, X, Y or Z, until he or she is satisfied. Throughout the process, the program keeps tracking the 3D coordinates of the cube.

In a user interactive system, it is reasonable to assume that a user can locate the objects at his or her will. However, as we know, the single image cannot provide depth information. That is, even though the cube looks like it is on the table from the front view, one might find out that the cube floats above the table from the side viewpoint. Inspired by stereo vision, the system in figure 3.4 gives a stereo view of the table. When the user manipulates the cube, the cube moves simultaneously in two views, and the user can obtain depth information by watching the cube from two different views.

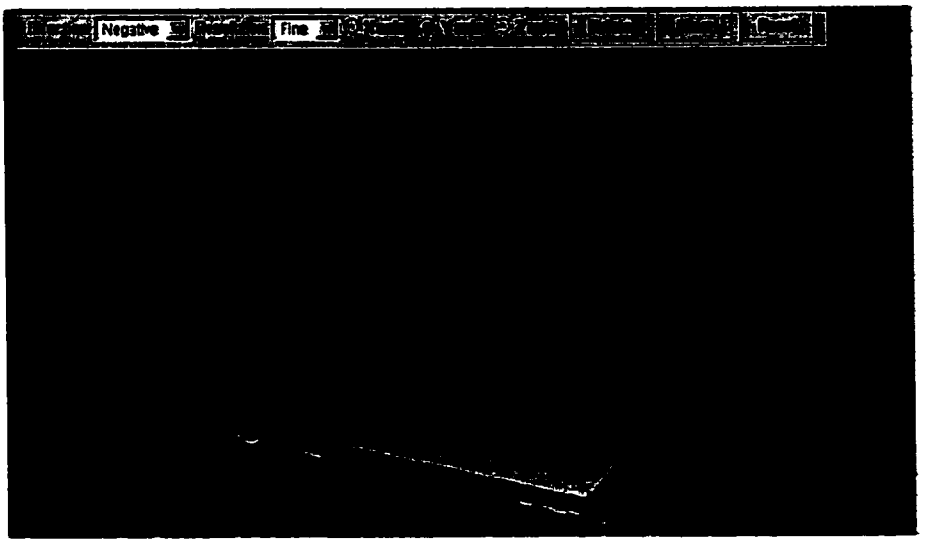

Figure 3.3: Using the manipulator to position the virtual object

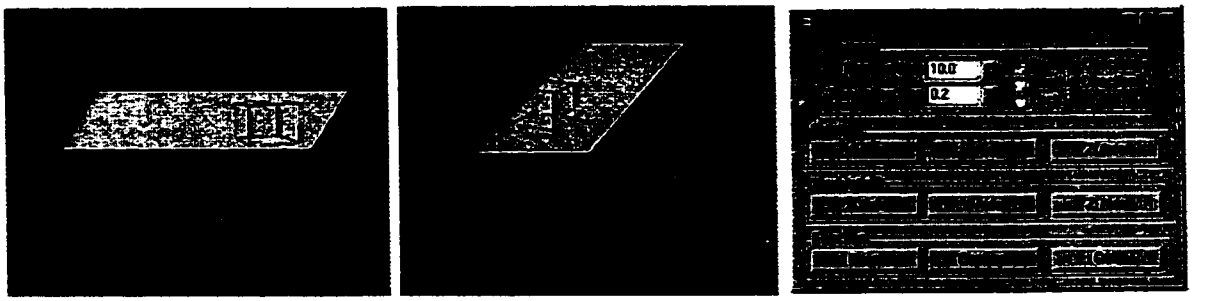

Figure 3.4: Positioning the virtual object with two views

A similar method has been reported in [10]. The approach is as follows: place a real object in the environment and attempt to register a virtual object with that real object. The user watches at one viewpoint or a few selected viewpoints and manually adjusts the location of the virtual object and the other viewing parameters until the registration looks right.

The use of the manipulator gives the user the flexibility to position the virtual object. Meanwhile the user has to be trained to operate the manipulator since it is still not a natural way to position an object. Furthermore, the disparity of the stereo view has to be large enough, otherwise the depth information cannot be shown clearly to the user. This, however, will create a problem since the large disparity of the view is not always easy to obtain in an image sequence.

3.1.3 Stereo Positioning

It would be a natural way to position an object by clicking the desired position using a mouse in a target image and the object is automatically placed on that location. Kutulakos et al. in [56] gives such an interactive scheme. The idea borrowed the result from stereo vision: given a point in space, its 3D location is uniquely determined by the point's projections in two images taken at different positions of the camera. That is, unlike the method of using the manipulator that moves the object to its place, they directly specify the location of the virtual object on a stereo image pair, as shown in figure 3.5.

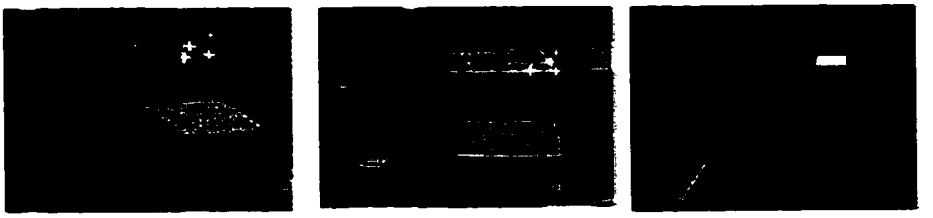

Figure 3.5: Stereo positioning

Actually only 3 points on the object are needed to determine the object's location and orientation in space. In the method, after the user specifies those 3 points on the left

image, constraints such as epipolar constraint, collinear and coplanar constraints are applied to help determine their location in the right image. For example, in figure 3.5, the goal is to align the cube's corner with the right corner of the workstation. The epipolar lines in right image and the sides of the workstation help to determine the cube's location in right image. This method moves a step forward to naturally position an object. However, it asks user to switch between the stereo images and need extra alignment information; for example, using the sides of workstation to help positioning.

3.2 Camera Pose Estimation

In order to simulate the real camera, we have to know its intrinsic and extrinsic parameters. The intrinsic parameters describe camera's optical, geometric and digital characteristics: the focal length f, optical center (o_x, o_y) and the digitalization factor (s_x, s_y). The extrinsic parameters describe its position and orientation with respect to the world reference frame: the camera position coordinates (x_c, y_c, z_c) and the rotational angle α, β, γ respect to world reference frame axes x, y and z.

Both intrinsic and extrinsic parameters can be estimated through a camera calibration process. However, calibrating the camera in real-time is not necessary. For many video-based augmented reality systems, the intrinsic parameters are fixed throughout the time. The intrinsic parameters only need to be calculated when the system is set up. What we need to dynamically estimate is the pose of the camera. That is, for each frame of the image stream, we will calculate the position and orientation of the camera so that the virtual object can be correctly rendered and blended using the real camera's viewpoint.

There has been much research that uses non-vision technology for pose estimation; for example, use of magnetic trackers or ultrasonic transmitters to report the position and orientation of user's (camera) viewpoint. However, the magnetic trackers are subject to large amounts of error and jitter, particularly in the presence of magnetic field disturbances such as metal and electric equipment.

In a video-based augmented reality system, video images of the user's view are always available. Therefore, it would be a reasonable approach to use those images

to track the camera's position and orientation. Camera tracking has been extensively investigated in the field of computer vision. Actually, vision-based tracking is possible to provide accurate data at long ranges. The tracking techniques are beyond the topic of discussion. Hence we will focus on the estimation of camera pose using the geometric information of tracked points and their 3D counterparts.

3.2.1 Algorithms for Location Determination Problem

Estimating the camera location from known space points is also known as location determination problem (LDP). It has been proved [31] that at least 3 space points are needed to have the finite solution for LDP. Figure 3.6 shows 3 points are seen by a single camera. Point C_p is the optical center of the camera. p_1, p_2 and p_3 are known space points. This is also known as the perspective 3-point problem (P3P).

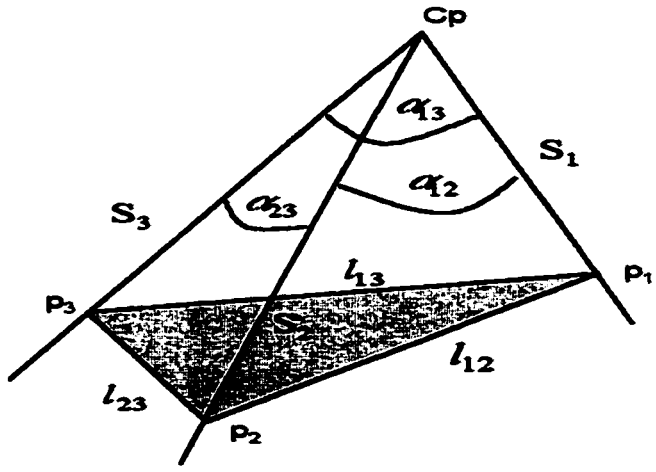

Figure 3.6: The P3P problem

From figure 3.6, the following equations hold:

$$\begin{aligned} l_{12}^2 &= s_1^2 + s_2^2 - 2s_1 s_2 \cos\alpha_{12} \\ l_{13}^2 &= s_1^2 + s_3^2 - 2s_1 s_3 \cos\alpha_{13} \\ l_{23}^2 &= s_2^2 + s_3^2 - 2s_2 s_3 \cos\alpha_{23} \end{aligned} \tag{3.1}$$

where s_1, s_2 and s_3 are the distance from point C_p to point p_1, p_2 and p_3; α_{ij} is the angle between s_i and s_j; $l_{12} = \|p_1 - p_2\|$; $l_{13} = \|p_1 - p_3\|$; $l_{23} = \|p_2 - p_3\|$.

The above polynomial equation set will have 8 solutions for each unknown s_1, s_2, s_3. However, because every term in the equation set is either a constant or of second degree, for every real positive solution there is a geometrically isomorphic negative solution [31]. Thus we obtain at most 4 positive solutions.

According to [31], for 4 or 5 known space points, a unique solution for LDP is not always assured; [31] gives an example that generates two possible solution for both 4 or 5 space points. However, if the 4 space points are co-planar, the unique solution is guaranteed.

For 6 or more known space points, LDP always has unique solution, since the projection matrix can be solved like camera calibration.

A P3P can be solved using geometry analysis. The normalized vectors j_i pointing from C_p to p_i are given by:

$$j_i = \frac{1}{\sqrt{u_i^2 + v_i^2 + f^2}} \cdot \begin{pmatrix} u_i \\ v_i \\ f \end{pmatrix} \tag{3.2}$$

where (u_i, v_i) is the projection of p_1, p_2 and p_3 in image plane. We can then get the equation:

$$\begin{aligned} \cos\alpha_{12} &= j_1 \cdot j_2, \\ \cos\alpha_{13} &= j_1 \cdot j_3, \\ \cos\alpha_{23} &= j_2 \cdot j_3 \end{aligned} \tag{3.3}$$

where α_{ik} is the angle between j_i and j_k.

Thus, s_1, s_2, and s_3 which are the distance from point C_p to point p_1, p_2 and p_3 can be solved by equation (3.1) and (3.3). Given the 3D coordinates of space points p_1, p_2 and p_3, and the lengths of the 3 legs s_1, s_2, s_3, the 3D location of C_p can be solved as follows [31]:

1. Knowing the side length s_1, s_2 and l_{12}, we can calculate angle $\angle C_p p_1 p_2$ by the law of cosines. Thus, we can know the projection of C_p on line $< p_1 p_2 >$, point Q. Next, construct a plane π_1 that is perpendicular to $\overrightarrow{p_1 p_2}$ and passes point Q. Apparently, plane π_1 also passes through point C_p.

2. Similarly, construct π_2 that is perpendicular to $\overrightarrow{p_1 p_3}$ and passes through C_p and Q'. Point Q' is the projection of C_p on line $< p_1 p_3 >$.

3. Construct plane π_3 defined by p_1, p_2, p_3.

4. Plane π_1, π_2 and π_3 have intersection at point R. By construction, vector $\overrightarrow{RC_p}$ is perpendicular to plane π_3.

5. Compute the length $\|p_1 R\|$ and use that in conjunction with s_1 to compute the length of vector $\overrightarrow{RC_p}$.

6. Computer the cross product of vector $\overrightarrow{p_1 p_2}$ and $\overrightarrow{p_1 p_3}$ to form a vector perpendicular to plane π_3. Then multiple that vector by length of $\|RC_p\|$ and add it to point R to get the 3D coordinates of C_p.

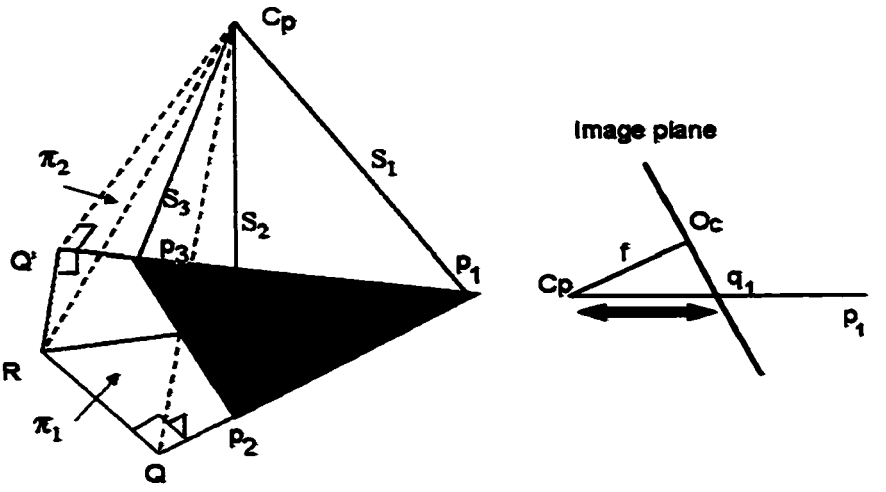

Figure 3.7: The geometric solution for the P3P problem

The camera's orientation is the orientation of its image plane. By knowing the projection of C_p on the image plane, point O_c and focal length f, it is easy to determine the orientation of the image plane:

1. Compute the vectors from point C_p to p_1, p_2, p_3 respectively, where the magnitude of these vectors are the distances from C_p to q_1, q_2 and q_3, the projections of p_1, p_2 and p_3 in image plane.
2. Add those vectors to C_p to get the 3D coordinates of q_1, q_2 and q_3.
3. Compute normal of image plane from q_1, q_2 and q_3. The normal presents the orientation of the camera.

3.2.2 Iterative Algorithms

Because of the image noise and tracking error, pose estimation using limited number of correspondences will not be very reliable. To make system robust against the image noise, we usually use more point correspondences, which makes the system over determined. Whenever the number of correspondences are larger than four, an iterative non-linear method is more efficient.

Projection from 3D to 2D is a nonlinear operation. However, it is a smooth and well-behaved transformation. Therefore, Newton's method can be applied. While given an appropriate initial estimation, we can get a good estimation of the pose parameters from the results of the last frame during tracking. That is, the current pose parameters are calculated by adding a correction vector X to the last estimation:

$$P_k + 1 = P_k + X \tag{3.4}$$

where vector X is calculated using the following equation:

$$\begin{aligned} JX &= E, \\ X &= (J^T J)^{-1} J^T E \end{aligned} \tag{3.5}$$

where E is a vector of measurements error, whose elements are x, y coordinates of errors between tracked feature positions and re-projected feature positions computed with the last pose estimation. J is the Jacobian matrix $J_{ij} = \partial x_{f_i}/\partial p_j$, where elements of x_f are x, y coordinates of feature points in the image and p_j is the pose parameters on each iteration.

Apply equation (3.5) to (3.4) iteratively with a reasonable initial guess until P is stable. In most cases, as long as there are many more measurements than parameters, Newton's method will usually converge in a stable manner from a wide range of starting positions.

Chen et al. [20] proposed a hybrid method, P3P-ICP. Instead of estimating the camera coordinates C_p in world reference frame, P3P gives the 3 seed points' coordinates in camera reference frame. As usual, it has multiple solutions. These solutions are used as an initial estimation for ICP(Iterative Closet Point) algorithm, and the solution with the least residue error is served as solution of the P3P-ICP approach.

The basic idea of the ICP algorithm is to first establish point correspondences by finding the closest neighboring points and then compute the rigid motion via a least-square error measure. The P3P-ICP method is reported very reliable and robust in Chen's experiments.

DeMenthon and Davis [26] attempted to use fast linear techniques, associated with the weak perspective camera model in order to obtain the pose that is associated with the perspective camera model. The method starts with computing the object pose using a weak perspective model and after a few iterations converges towards a pose estimated under perspective.

Over all, nonlinear iteration approaches have two drawbacks:

- Need for good initial estimation of the true solution.
- It is a time consuming computation process.

3.2.3 A Novel P4P Method

A novel and practical camera pose estimation algorithm is proposed using 4 co-planar points [81]. As discussed in previous section, 4 co-planar points(P4P) can uniquely determine the pose of the camera.

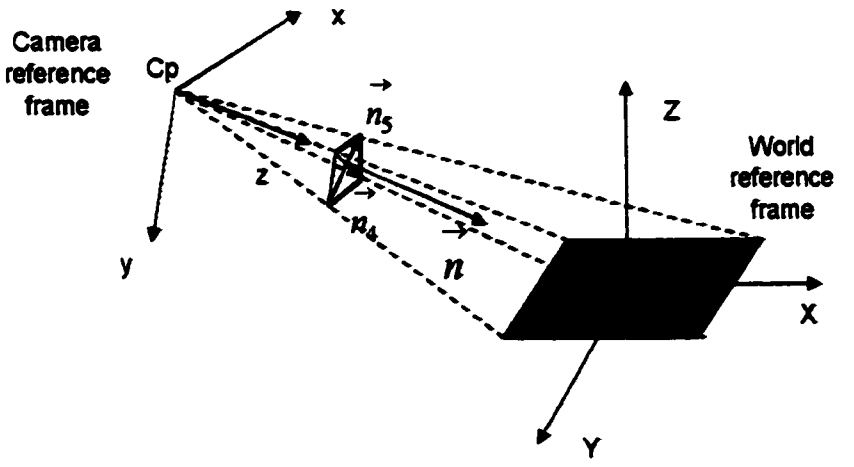

Figure 3.8: A novel P4P method

The method tracks 4 corners of a square landmark, as shown in figure 3.8. The world reference frame is assigned at the center of the landmark so that the matrix of the transformation from world reference frame to camera reference frame is easy to be configured:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{e_x} & \vec{e_y} & \vec{e_z} & \vec{e_t} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.6}$$

where e_x, e_y and e_z represent the rotation coefficients, and e_t represents camera motion,

as:

$$\begin{aligned} \overrightarrow{e_x} &= N(\overrightarrow{v_5} - \overrightarrow{v_4}), \\ \overrightarrow{e_y} &= N(\overrightarrow{v_4} + \overrightarrow{v_5}), \\ \overrightarrow{e_z} &= N(\overrightarrow{n}), \\ \overrightarrow{e_t} &= dist \times N(\overrightarrow{p}) \end{aligned} \tag{3.7}$$

where *dist* represents the distance from camera center to the center of the landmark, vector $\overrightarrow{p}$ is the vector from camera center to the center of the landmark on the image plane, and $N(\overrightarrow{v})$ is a normalization function. $\overrightarrow{v_4}$ and $\overrightarrow{v_5}$ are the diagonal vectors of the tracked landmark in the image plane. $\overrightarrow{n}$ is the normal vector of the image plane.

This method is neat, and the transformation matrix can be directly applied for projecting object in world reference frame to camera image plane. This method has been successfully implemented in a head-mounted and hand-held augmented reality system [81].

In summary, camera pose estimation is a primary step to register a virtual object with real scene. The pose estimation techniques have evolved from the early techniques that use magnetic sensors to currently vision-based pose recovering algorithms.

3.3 Virtual Camera Simulation

Some existing graphics application programming interfaces(API), for example OpenGL, can be used to model and render the 3D virtual object. The virtual camera thus can be simulated by OpenGL using a set of camera parameters obtained through camera calibration and pose estimation.

OpenGL is a cross-platform graphics API that is widely used in graphics society. OpenGL has a well-defined specification in which a camera model is defined. Figure 3.9 shows the image formation pipeline in OpenGL.

The ModelView matrix first transforms the object coordinates(world reference frame) to eye coordinates (camera reference frame). After that, OpenGL applies the Projec-

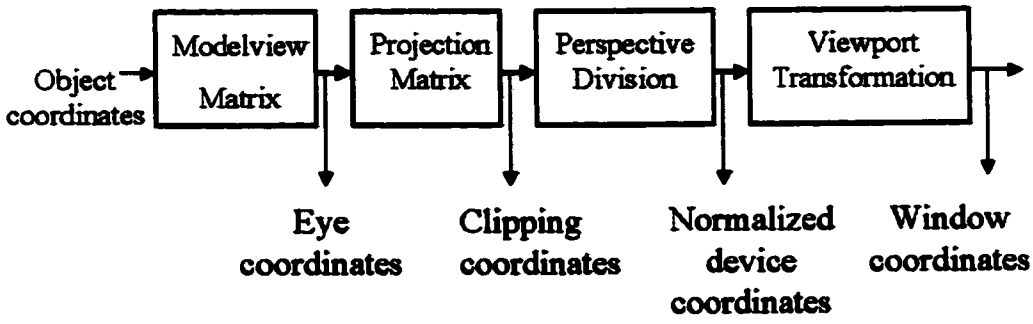

Figure 3.9: OpenGL image formation pipeline

tion matrix to yield clip coordinates. This transformation defines a viewing volume; objects outside this volume are clipped so that they're not drawn in the final scene. After this point, the perspective division is performed by dividing coordinate values by w (which is introduced by homogeneous coordinates), to produce normalized device coordinates. Finally, the transformed coordinates are converted to window coordinates by applying the viewport transformation. In [68], Li Ming gives a correspondence analysis between the OpenGL's image formation pipeline and the camera calibration using a pinhole model. Given an image with its dimension and its recovered camera parameters t_0, t_1, t_2, α, β, γ, f, u_c, v_c and dx/dy, we can present the viewport, projection and ModelView matrix setup in turn:

1. Viewport transformation:

$$glViewport(u_c - width/2, v_c - height/2, width, height); \tag{3.8}$$

2. Projection transformation:

$$gluPerspective(2 * \arctan(\frac{height}{2} / \frac{\lambda * f}{dy}) / M\_PI * 180, \frac{width}{height} * \frac{dx}{dy}, 0.1, 200.0); \tag{3.9}$$

3. ModelView transformation:

$$\begin{aligned} &glTranslatef(t_1, t_2, t_3); \\ &glRotatef(\alpha/M\_PI * 180, 0, 0, 1); \\ &glRotatef(\beta/M\_PI * 180, 0, 1, 0); \\ &glRotatef(\gamma/M\_PI * 180, 1, 0, 0); \end{aligned} \tag{3.10}$$

Thus, with these equations, OpenGL can be used to simulate the virtual camera with the parameters obtained from the camera calibration.

3.4 Camera Calibration Free Method

Recently, another class of AR approaches which use affine spaces for integration of virtual objects was proposed [56]. These approaches are referred to as calibration free approach, which means the projection matrix used for projecting the virtual object is not obtained by a camera calibration process. The algorithm is based on two affine properties:

- Affine reprojection property

  The projection of a 3D point in any new image being viewed by the camera is a linear combination of the projections of the affine basis points:

$$\begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} = \begin{pmatrix} u_{p_1} - u_{p_0} & u_{p_2} - u_{p_0} & u_{p_3} - u_{p_0} & u_{p_0} \\ v_{p_1} - v_{p_0} & v_{p_2} - v_{p_0} & v_{p_3} - v_{p_0} & v_{p_0} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.11}$$

  where $(x, y, z, 1)^T$ is the homogeneous affine coordinates of space point p. $(u_{p_i}, v_{p_i}, 1)^T$ is the projection of the affine basis, and $(u_p, v_p, 1)^T$ is the projection of point p.

- Affine reconstruction property

The affine coordinates of any point can be computed from equation 3.12 if its projections in at least two views are known and the projections of the affine basis points are also known in those views.

$$\begin{pmatrix} u_p^1 \\ v_p^1 \\ u_p^2 \\ v_p^2 \end{pmatrix} = \begin{pmatrix} u_{p_1}^1 - u_{p_0}^1 & u_{p_2}^1 - u_{p_0}^1 & u_{p_3}^1 - u_{p_0}^1 & u_{p_0}^1 \\ v_{p_1}^1 - v_{p_0}^1 & v_{p_2}^1 - v_{p_0}^1 & v_{p_3}^1 - v_{p_0}^1 & v_{p_0}^1 \\ u_{p_1}^2 - u_{p_0}^2 & u_{p_2}^2 - u_{p_0}^2 & u_{p_3}^2 - u_{p_0}^2 & u_{p_0}^2 \\ v_{p_1}^2 - v_{p_0}^2 & v_{p_2}^2 - v_{p_0}^2 & v_{p_3}^2 - v_{p_0}^2 & v_{p_0}^2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.12}$$

where $(u_p, v_p, 1)^T$ and $(u_{p_i}, v_{p_i}, 1)^T$ are the projections of point p and affine basis point p_j in image I_i respectively, .

The following is the outline of the calibration free algorithm for a video-based augmented reality system :

1. The user selects four corresponding fiducial points in the stereo image pair to establish the affine basis.
2. Projection matrix is then calculated using affine reprojection property.
3. The user selects four non-coplanar vertices $p_1, ...p_4$ on the 3D virtual object, and specifies the projections of $p_1, ...p_4$ in the stereo image pair.
4. Compute the affine coordinates using affine reconstruction property.
5. Compute the affine coordinates of all points on the model of virtual object from the affine coordinates of $p_1, ...p_4$.
6. The affine basis is tracked when camera is moving. The projection matrix is dynamically updated and the virtual object is dynamically overlayed on the real scene's video stream.

The use of an affine framework for formulating the video overlay problem is both a strength and a limitation of the calibration free augmented reality approach. The approach gets rid of the error prone and highly computational cost process, the camera calibration. On the other hand, the approach has some drawbacks [56]:

- Relies on an affine approximation to perspective projection. This restricts system operation to relatively large object-to-camera distances, for example, greater than 10 times the object's size.
- Ignores radial camera distortion.
- Uses purely nonmetric quantities to render virtual objects. Rendering techniques that require metric information are not directly supported, for example, angle measurements for lighting calculations.
- Relies on point tracking to generate the live video overlay.

Over all, calibration-free augmented reality is a good attempt to build an efficient, portable augmented reality system. It gets rid of camera calibration process. Hence, it is particularly suitable in the situation that the camera focal length is dynamically changed.

3.5 Dynamic Registration Correction

So far, the registration procedure is still an open loop procedure. Figure 3.10 shows the transfer function diagram. There are two inputs, the 3D virtual object and the image stream captured by a real camera. The camera's extrinsic parameters (the camera pose) and intrinsic parameters are recovered by the camera pose estimation module and the camera calibration module. Once the virtual object is positioned in the real scene, it is rendered by a simulated virtual camera and the rendering is merged with the real image stream. In other words, the registration procedure is actually a series of transformations. Given a 3D virtual object, it is first transformed to the desired position with a desired orientation in the world reference frame. Then the virtual object is transformed to the camera reference frame. At last it is projected onto the camera image plane.

To achieve a better registration, the only way for this open loop scheme is to make each component(transformation) more accurate. Some success at improving registration error has been achieved with auto calibration approaches [35] and predictive

tracking techniques [10]. However, they all suffer from the drawbacks due to the open loop scheme, such as, sensitive to noises and signal drift.

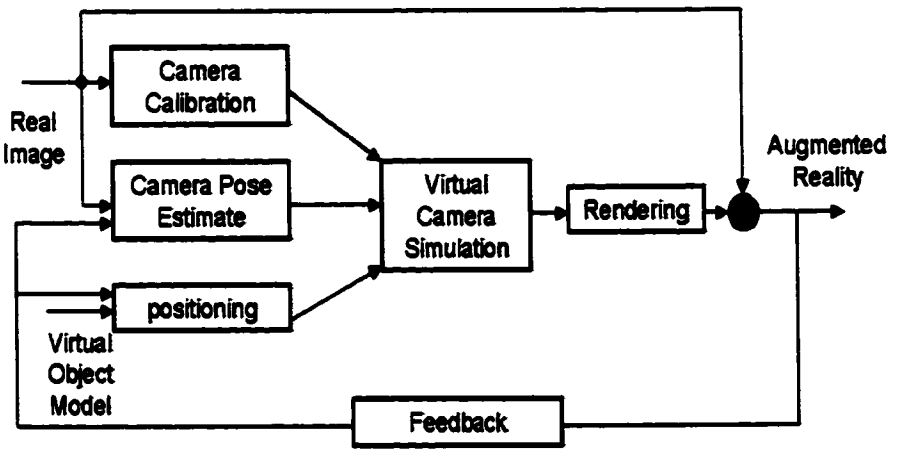

Figure 3.10: Transfer function diagram of an augmented reality system

Alternatively, a close loop system senses its own output and corrects errors dynamically. A close loop AR system dynamically measures the 2D registration error in combined images and uses that information to correct 3D coordinate system's registration error, which makes the system tolerant of transformation inaccuracies.

It is difficult to determine which particular errors are causing misregistration. The selection of which transformation parameters to adjust depends on both their uncertainty and how sensitive image-space errors are to that uncertainty. For example, for objects far from the camera, the orientation error of camera pose causes more misalignments than the position error; while for objects near the camera, the situation is inverted.

Two approaches to reduce registration errors are studied in [12]. One assumes that the camera pose and the Camera-to-Image transformation is correct, and the registration correction is to dynamically adjust the position of the virtual camera. Since the registration metric gives no estimate of distance between each object and the camera, virtual objects are displaced on a constant radius from the virtual camera's

viewpoint. Thus, it will not cause the virtual object shrinking or growing unnaturally.

The other approach assumes that every transformation is correct except the camera pose transformation. That is, the camera pose will be adjusted dynamically to achieve minimal registration error. For a further simplification, only camera orientation is adjusted. This is because in most cases, the object is far from the camera and the registration errors are more sensitive to the camera orientation error. Figure 3.11 shows how registration error can be eliminated using this approach.

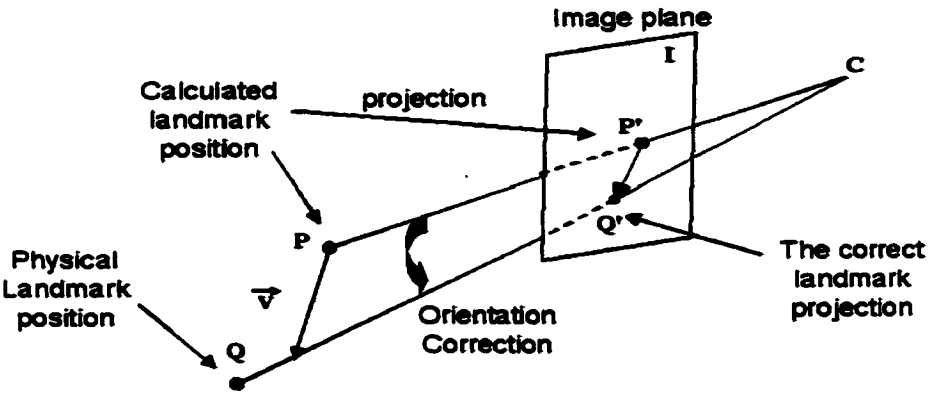

Figure 3.11: Dynamic registration correction

In the figure 3.11, point C and Q are the camera position and the landmark position in the physical 3D space respectively. I is the image plane. The projection of landmark on the image, Q' is the intersection point of line $< CQ >$ and the image plane I.

Because of the sensor error, the landmark's position is reported as P which is different from the real position Q. This causes a registration error: the landmark's projection shifts from Q' to P'. To correct the shift, we will translate P by vector $\vec{v}$ so that its projection P' coincides with point Q' on the image:

$$\vec{v} = n(\overrightarrow{CQ'} - \overrightarrow{CP'}) \tag{3.13}$$

where n is a scale factor derived by:

$$n = \| CP \| / \| CP' \| \tag{3.14}$$

Since this method only registers the positions on 2D image plane, the three dimensional position may not coincide even after the correction. However, it is still effective if the sensor error is not big.

3.6 Conclusion

The positioning of virtual object is a user-computer interactive procedure and should be both accurate and user-friendly. All the positioning methods we discussed in this chapter have their limitations and application domains, for example, the positioning using the manipulator in section 3.1.2 is used in medical diagnosis systems.

The pose estimation is a computationally expensive process. The accuracy of pose estimation largely depends on the accuracy of the tracking system. Tracking techniques may be grouped into three categories:

- Active target tracking. Active target tracking uses powered signal emitters and sensors. For example, magnetic, optical, radio, and acoustic signals.
- Passive target tracking. Passive target tracking uses naturally occurring signals, for example, the intentionally placed fiducial or natural features used in vision-based tracking.
- Inertial tracking. Inertial tracking uses inertial sensors that sample the linear acceleration and angular motion.

To achieve robust tracking for registration, research has been in two different directions. The first is based on pure vision-based tracking, and the other is based on hybrid sensing tracking. Though computer vision technique provides an accurate measurement of geometry for augmented reality, it is proved not robust enough. As we have seen, both calibration intense and calibration free augmented reality system need to track a few fiducial points. In nature, the fiducial points can be occluded, obscured, or may disappear from the camera's view at any time. Failure to track these fiducial points will simply break down the system. An improvement is that instead of simply tracking some isolated points, the system tracks lines, curves or circles [22]. This will

be a more robust tracking, since though you might miss a point, you could not miss the whole pattern.

Hybrid systems attempt to compensate for the shortcomings of each technology by using multiple measurements to produce robust results. In [82], the robust magnetic tracking is used to provide a reliable estimation of the camera pose, and then the vision-based tracking is used to refine this estimation. In [102], hybrid tracker fuses inertial orientation (3DOF) data with vision feature tracking to stabilize performance and correct inertial drift. The inertial data also serves as an aid to the vision tracking by reducing the search space and providing tolerance to interruptions.

To improve the registration accuracy, one can use close loop feedback by measuring image registration error. The feedback can compensate the tracking errors and lens distortions exist in optical system. Therefore, the close loop correction of registration will be a trend in future augmented reality systems.

Chapter 4

Interactive Virtual Object Registration

Our approach of registering 3D virtual objects onto the real scene is based on 3D reconstruction. In this approach, the user is provided a pair of calibrated stereo images and a view of the 3D virtual object. The user can choose either images to register the virtual object with. Before the registration, the program will reconstruct the 3D model of the plane where the virtual object is going to be registered. After that, the user can click on the registration plane on the image to specify the virtual object's location, and the location's 3D coordinates are immediately calculated. The virtual object is then positioned by a simple transformation.

The registration process can be outlined as follows:

1. Prepare the stereo image. The target real environment is taken by two cameras and the projection matrices are known by prior camera calibration processes.

2. Define the registration plane. The user specifies 3 points by clicking the mouse on one image. The corresponding points on the other image will automatically be matched using epipolar constraint and similarity matching. Thus, the plane is reconstructed using the 3 pairs of stereo points.

3. Position the virtual object. The user selects a vertex from the virtual object and

specifies its projection on the reconstructed plane by clicking the mouse on the image. Only 3 vertices are needed to determine the position and orientation of the virtual object. Then, the virtual object is transformed and projected onto the image of the real environment.

4.1 Semi Automatic Camera Calibration Tool

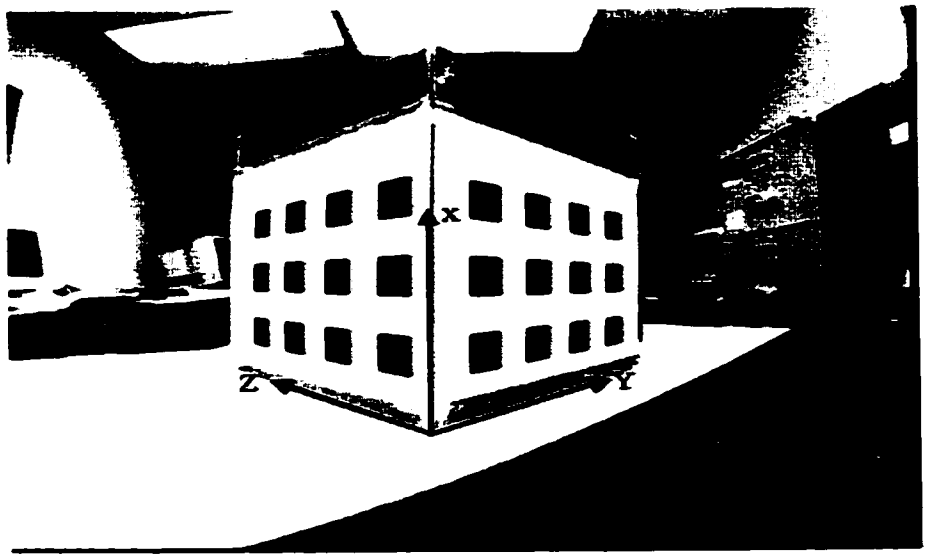

Figure 4.1: The coordinates of the calibration pattern

Our method for registration is based on 3D reconstruction from calibrated images. The camera calibration is to use a set of space points (feature points) and their projections, the image points to recover the projection matrix. In our method, a calibration pattern is used, as shown in figure 4.1. The calibration pattern consists of two orthogonal planes on which many black squares are located. The feature points are the corners of black squares so that their 3D coordinates can be obtained easily. For example, we make the left plane as XZ plane and the right plane as YX plane. The coordinates of the corners of the black squares can be measured on two planes. The problem that remains is how to extract the feature point's projection on the image grabbed by the camera and how to build the correspondence with each other. Our semi automatic calibration

method will use corner detection to extract the pixel coordinates of those corners and use homography to build the correspondences.

4.1.1 Homography

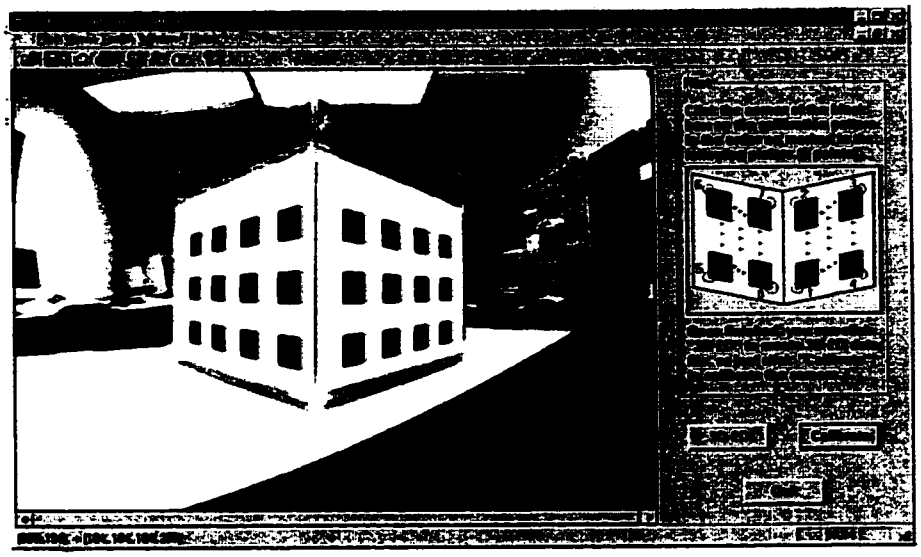

Figure 4.2: Screen snapshot of the semi automatic calibration

Figure 4.2 is the screen snapshot of our semi automatic calibration interface. The image shows the calibration pattern grabbed by the camera. Meanwhile the data of 3D coordinates of the feature points are also loaded from the data file into the program. The feature points of calibration pattern on XY plane and XZ plane are projected onto the image plane by the plane projective transformation, also known as plane to plane homography. Equation 4.1 describes the transformation:

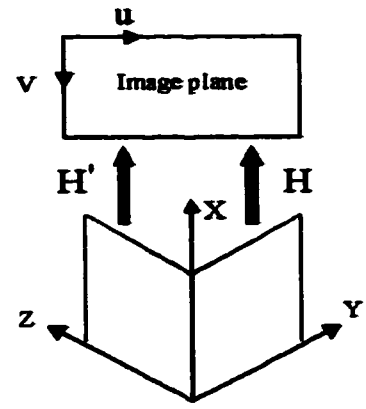

Figure 4.3: Homography transformation between pattern plane and image plane

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} h'_{11} & h'_{12} & h'_{13} \\ h'_{21} & h'_{22} & h'_{23} \\ h'_{31} & h'_{32} & h'_{33} \end{pmatrix} \begin{pmatrix} x \\ z \\ 1 \end{pmatrix} \quad (4.1)$$

where (u, v) is the pixel coordinates, and (x, y), (x, z) are coordinates for points on plane XY and XZ respectively. Note that the homography matrix has 9 entries, but only 8 are independent. Hence by setting $h_{33} = 1$ and $h'_{33} = 1$, the homography matrix can be solved using SVD if we know 4 points on the pattern plane and their corresponding projections on the image.

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1x_1 & -u_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -v_2x_2 & -v_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -v_3x_3 & -v_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -u_4x_4 & -v_4y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -v_4x_4 & -v_4y_4 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix} \tag{4.2}$$

$$\begin{pmatrix} x_1 & z_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1z_1 \\ 0 & 0 & 0 & x_1 & z_1 & 1 & -v_1x_1 & -u_1z_1 \\ x_2 & z_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2z_2 \\ 0 & 0 & 0 & x_2 & z_2 & 1 & -v_2x_2 & -v_2z_2 \\ x_3 & z_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3z_3 \\ 0 & 0 & 0 & x_3 & z_3 & 1 & -v_3x_3 & -v_3z_3 \\ x_4 & z_4 & 1 & 0 & 0 & 0 & -u_4x_4 & -v_4z_4 \\ 0 & 0 & 0 & x_4 & z_4 & 1 & -v_4x_4 & -v_4z_4 \end{pmatrix} \begin{pmatrix} h'_{11} \\ h'_{12} \\ h'_{13} \\ h'_{21} \\ h'_{22} \\ h'_{23} \\ h'_{31} \\ h'_{32} \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix} \tag{4.3}$$

In our method, these 4 points are the extreme corners on XY and XZ pattern plane, as shown in figure 4.2. The user is asked to click on these extreme corners on the image using mouse to obtain their pixel coordinates. The clicking is in a sequential order(see figure 4.2) so that the correspondences between the extreme corners and their projections are also built.

The pixel coordinates calculated by homography will not exactly match those square corners on the image. This is because the user is unable to accurately click on the extreme corners on the image. Furthermore, the image might also be distorted by camera's lens. In practice, we only ask the user to click the extreme corners as close as he or she can, and the pixel coordinates calculated by homography are only used for reference. A corner detector will refine the coordinates accurately and finally build the 3D-2D correspondences for calibration.

4.1.2 Image Corner Detection

Feature detection is widely researched in computer vision. In computer vision, an edge is defined as a one-dimensional discontinuity of image brightness function. A corner is defined as a two-dimensional discontinuity of the image brightness function. Therefore, corners in an image are easier to be detected and tracked through the sequence of images when the camera moves and lighting conditions of the scene change.

An ideal corner in an image has an infinite sharpness and a clear boundary. However, in practice, corners are typically rounded, blunted, blurred and ragged. Furthermore, as corners in image are detected based on image intensity discontinuities, the intensity discontinuities can be caused by surface discontinuity in 3D space, texture changes on the objects or by some shadows cast by other objects etc., and therefore it may not be associated with a corner in a real scene, which is a junction of three or more surfaces.

Many corner detectors have been developed recently. They mainly fall into two catalogues: template-based corner detector and geometry-based corner detector. Template-based corner detector first develops a set of corner templates and then performs the similarity matching between the templates and all the searching windows of a gray level image. This kind of corner detector is not widely used because of its high algorithm computational cost. Geometry-based corner detector relies on measuring differential geometry features of corners. Some of them are widely used in all kinds of applications.

Given a digital image, the early geometry-based corner detectors extract edges as a chain code, and then search for points having maximal curvature. The computational cost of this kind of method is still high, and any errors in the segmentation step will lead to great astray in the corner results. Later researches change to directly operate on the gray level image. However, most of them are using gradient information and therefore sensitive to noises.

Smith and Brady introduced a straightforward method named SUSAN that does not depend on image derivatives. Given a digital image, the USAN (Univalue Segment Assimilating Nucleus) area will reach a minimum when the nucleus lies on a corner point. SUSAN is not sensitive to noise and is very fast as it only uses very simple

operations. As in figure 4.4, all pixels within a circular mask are compared with the nucleus, using the comparison equation:

$$n = \sum_i 100 \cdot e^{-(\frac{I_i - I_c}{t})} \tag{4.4}$$

where I_c is the intensity of the nucleus that is being examined and I_i is the pixel intensity in the mask, and t is the threshold. It determines the maximum difference in grey levels between two pixels which allows them to be considered part of the same region in the image. Thus, the pixel in the mask is grouped: it either belongs to the group of nucleus(USAN), or not. If the nucleus pixel is considered a corner candidate, the pixel number of its group must less than half of total pixel number in the mask, in our case, the number is 18. Equation 4.5 describes this selection:

$$r = \begin{cases} n & \text{if } n < 0.5 n_{max}; \\ 0 & \text{if } n \geq 0.5 n_{max}, \qquad n_{max} = 37; \end{cases} \tag{4.5}$$

where r represents the response of the nucleus. If $r > 0$ then the nucleus is a corner candidate, and the corner candidate with locally minimal r value will be reported as a corner. This is intuitive because the r value represents the sharpness of the corner. The smaller the r value, the shaper the corner is. For further improvement against the image noise, SUSAN measures the gravity center of USAN and if the nucleus is too close to the gravity center, it will not be reported as a corner.

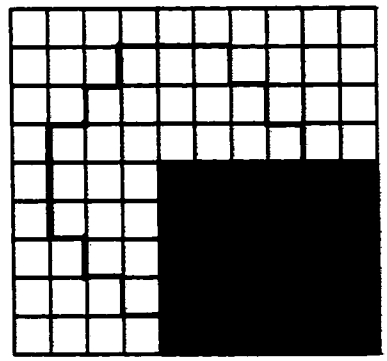

Figure 4.4: SUSAN corner detector

4.1.3 Camera Calibration

Once the image coordinates of the corner are identified and the correspondences between image points and space points are established, a camera calibration routine is called to calculate the projection matrix. In our method, the Leveberg-Marquardt nonlinear minimization method is used to solve the equations.

Throughout the calibration process, the user only clicks 8 points to help compute the homography, and the rest process is automatically performed by the program. The semi automatic calibration tool simplifies the user's operation and makes it easy to use, even for the user without the knowledge of camera calibration.

4.2 3D Reconstruction of the Registration Plane

Before registering a 3D virtual object, we first reconstruct the 3D model of the registration plane. As we know, a space plane is defined by 3 space points. The main idea behind the reconstruction is to recover the 3D coordinates of 3 space points from a pair of calibrated stereo images. The reconstruction is made as an interactive process:

1. The user picks 3 points on one of the stereo images to define the registration plane and the program records their pixel coordinates.
2. These 3 points are automatically matched on the other image by the program using similarity matching.
3. These three pixel points and their corresponding matching are used to reconstruct the 3D coordinates of their 3D counterparts, the space points.
4. The registration plane is finally reconstructed using the 3 space points.

In this interactive process, the user's operation is made simplest, only picking three points on one image by clicking the mouse.

```
Begin
highest(p_m)= 0;
compute the epipolar line l_p on the right image;
for each point p_i on the epipolar line l_p {
      if (shift(p_i, p) > 100) continue;
      open a searching window w_i centered at p_i;
      for each point p^k_{w_i} inside the window w_i {
            compute ZNCC(p^k_{w_i});
            if(highest(p_m)<ZNCC(p^k_{w_i}))
            { highest(p_m)=ZNCC(p^k_{w_i});
            p_m = p^k_{w_i};
            }
      }
}
point p_m is the matching;
End.
```

Table 4.1: Pseudo code for similarity matching

4.2.1 Stereo Matching

Once the user clicked a pixel on one of the stereo images, the program will automatically match this point on the other image. By applying the epipolar constraint, the searching area is narrowed along the epipolar line. That is, each pixel on the epipolar line is scored by ZNCC and the pixel gets the highest score will be considered a matching.

In order to know the epipolar line, we need to calculate the fundamental matrix. From section 2.4, we know that at least 8 pairs of pixel points on the stereo images are needed to solve the fundamental matrix. Recall that in the calibration of the stereo image pair, we know the pixel coordinates of the feature points on the calibration pattern in two images. Here we use these pixel points to calculate the fundamental matrix by equation 2.14.

Considering the noise and calculation error, matching of the selected pixel in the stereo image will not be exactly restricted along the epipolar line. Instead, for each point on the epipolar line, we open a small window and pixels inside the window are all examined by ZNCC. In addition, in order to improve the reliability and the speed,

not the whole epipolar line is searched. The fact is that the disparity of the two stereo images is usually not very large. Therefore the matching in one image will not shift very large from the one in the other image. In our program, this shift is made maximum 100 pixels. Table 4.1 is the pseudo code of matching left image point p on the right image.

4.2.2 Plane Reconstruction

The three space points' projections on the stereo images that allocated by the user and the similarity matching are then used to recover their 3D coordinates using the algorithm described in chapter 2. Thus, the reconstruction of the registration plane from these 3 space points is a straightforward process. That is, we calculate the coefficients a, b, c and d that define the plane equation: $ax + by + cz + d = 0$ by using following formula:

$$a = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix}, b = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}, c = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, d = - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} \quad (4.6)$$

where (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3) are the coordinates of the space points.

The reconstructed 3D coordinates will not be error free due to pixel noise and calibration error. This inaccuracy certainly will be propagated into plane reconstruction. In order to minimize such propagation, the disparity of the 3 space points should be as large as possible. Figure 4.5 shows how space points error and disparity will affect the reconstruction. Suppose the real plane Π is going to be reconstructed, and due to some errors, the reconstructed space points shift a certain value from the real. The plane defined by points with small disparity will give bigger discrepancy against the real plane than the plane defined by space points with large disparity.

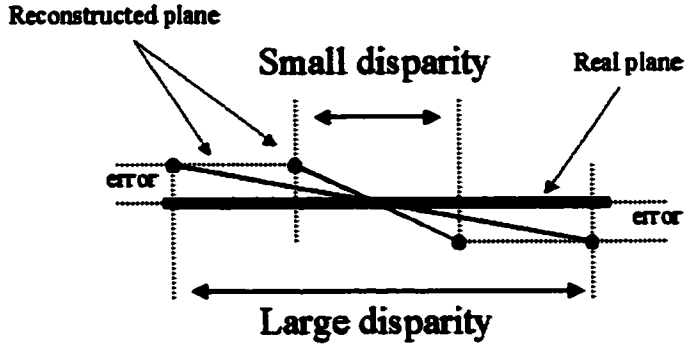

Figure 4.5: The plane modelling error and points disparity

4.3 3D Virtual Object Registration

In our system, we make the registration plane and the virtual object in the same coordinates system so that the estimation of the transformations for the virtual object is greatly simplified.

To register a 3D virtual object on the registration plane, the user picks a vertex from the 3D model of the virtual object and then specifies a location on the registration plane by clicking on the image. The program will recover the 3D coordinates of the new location by equation set 4.7:

$$\begin{aligned} u &= \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\ v &= \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\ 0 &= aX + bY + cZ + d \end{aligned} \tag{4.7}$$

where (u, v) is the pixel coordinates of the desired location. (X, Y, Z) is the 3D coordinates of this location. m_{ij}, $i = 1...3$, $j = 1...4$ are the entries of the projection matrix. $aX + bY + cZ + d = 0$ is the equation of registration plane we reconstructed. When the user specifies the second vertex on the registration plane, the user is actually assigning the direction of the second vertex on the registration plane. That is, the program will calculate the new location of the second vertex on the registration plane along the

line from the first assigned location to the second assigned location, according to the distance between these two vertices on the virtual object. This is because the user does not know the distance between the vertices on the virtual object, nor the scale between the virtual object and the registration plane. The third assignment will determine the up vector of the virtual object on the registration plane. When the user picks 3 vertices from the virtual object, the 3 vertices actually define a plane and the picking sequence determines the up vector(the normal). Similarly, the 3 new locations specified by the user also gives an up vector, which will be the up vector of the registered virtual object.

In summary, the user performs a registration by picking 3 vertices from the virtual object and clicking 3 points on the registration plane to define the position and orientation of the virtual object on the registration plane. Figure 4.6 shows the registration process.

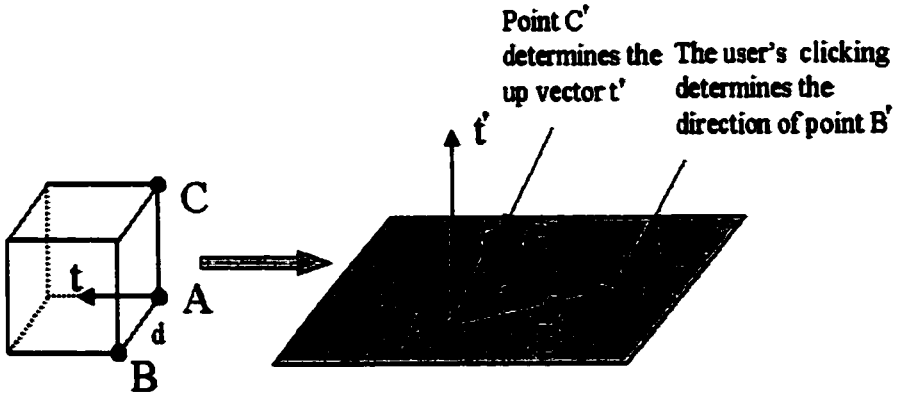

Figure 4.6: Registering virtual object using 3 points

Once the assignment is finished, the 3D virtual object will then be transformed onto the new location with the orientation as specified. The transformed 3D virtual object finally is projected onto the image of the real scene by the projection matrix obtained from camera calibration.

Figure 4.7 shows the sequential transformation of the registration. The plane Π_{ABC} is first rotated so that it's up vector t is parallel to the up vector t' of plane $\Pi_{A'B'C'}$ with the same direction. This can be done by rotating Π_{ABC} θ degree around vector

$\vec{m}$, where:

$$\begin{aligned} \vec{m} &= \vec{t} \times \vec{t'}, \\ \theta &= \arccos\left(\frac{\vec{t} \cdot \vec{t'}}{\| t \| \| t' \|}\right) \end{aligned} \tag{4.8}$$

Then, the virtual object is translated so that vertex A coincides with vertex A'. In order to make vertex B coincide with vertex B', the virtual object is rotated α degree about vector $\vec{t}$, where α can be calculated as:

$$\alpha = \arccos\left(\frac{\overrightarrow{AB} \cdot \overrightarrow{A'B'}}{\| AB \| \| A'B' \|}\right) \tag{4.9}$$

Once the vertices A, B and C are completely transformed to vertices A', B' and C', the same transformation will be applied to the other points of the virtual object. As a result, the virtual object is transformed to its new location. The program will finally project the virtual object onto the image using the projection matrix obtained from camera calibration.

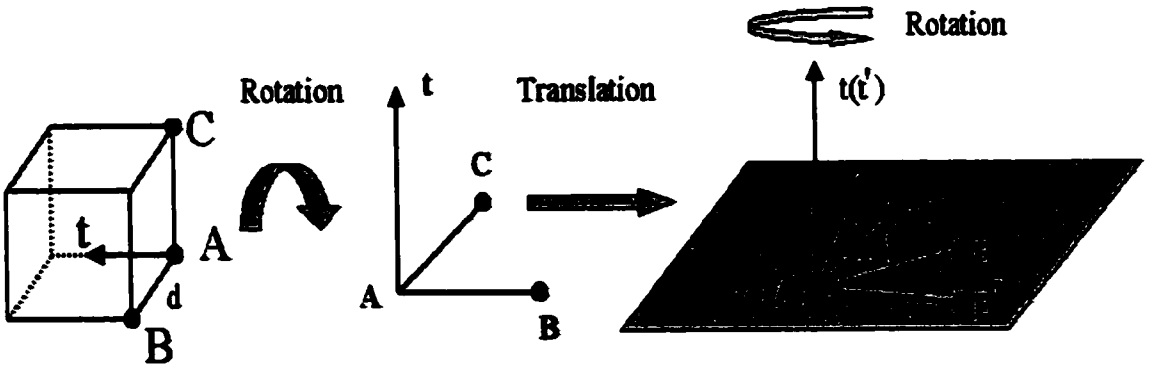

Figure 4.7: Transforming virtual object to the registration plane

Chapter 5

Implementation and Experiments

The method developed for our augmented reality system is implemented in Microsoft Visual C++ with MFC (Microsoft Foundation Classes). Visual C++ is a powerful and complex tool for building 32-bit applications for Window 95 and Windows NT. With its code-generating wizards, Visual C++ can produce the graphic user interface of a working Windows application in seconds. The class library included with Visual C++, the MFC, has become the industry standard for Windows software development in a variety of C++ compilers. The visual editing tools make a WYSIWYG (What You See Is What You Get) for programming layout of menus and dialogs. The 3D virtual object rendering and manipulation is programmed with OpenGL graphics library under Visual C++ frame work. All of the image processing in the program uses the Microsoft Vision Software Development Kit (Vision SDK), which is freely available for downloading from the Microsoft Research web pages.

5.1 Windows Programming

5.1.1 Messages

The use of messages sets windows programming apart from other kinds of programming. That is, everything that happens in a windows program is mediated by messages. A

message is a way for the operating system to tell an application that something has happened, for example, the user has clicked the mouse, or the printer has become available.

The operating system uses integers to refer to messages. However, for convenience, windows program refers to them by their names, and an enormous list of definition statements connects names to numbers. For example, message $0X000F$ has name of WM_PAINT. Different messages are handled by different parts of the operating system or the application. The heart of any windows program is the message loop, typically contained in a WinMain() routine. The WinMain() routine is, like the main() in DOS or UNIX, the function called by the operating system when you run the program. The WinMain() routine gets the messages from the operating system and dispatches them to proper massage handlers.

When you use MFC AppWizard to create your application, the AppWizard will generate the WinMain() routines for you. Hence, you will simply write the function that will handle the message, and add a message map to your class. Message maps come in two parts: one in the .h file for a class and one in the corresponding .cpp file. The entry added to the header file's message map declares the prototype of the handler function, and the entry in the source file's map associates messages with their handlers. AppWizard constructs the message map for most of the common messages and handle them properly. You will only add entries yourself.

5.1.2 View-document Paradigm

When an application runs under Microsoft Windows, the user interacts with documents displayed in frame windows. The MFC framework uses frame windows to contain views. A frame window class manages the frame, and a view class manages the contents. The view window is a child of the frame window. Drawing and other user interaction with the document take place in the view's client area, not the frame window's client area. The frame window provides a visible frame around a view, complete with a caption bar and standard window controls such as a control menu, buttons to minimize and maximize the window, and controls for resizing the window. The contents consist of

the window's client area, which is fully occupied by a child window - the view.

The MFC uses a document-centric approach for application design, which logically separates an application's data from the way the user actually views and manipulates that data, known as view-document paradigm. A document is an object that is responsible for storing, loading, and saving the data; whereas the view object enables the user to see the data on screen and to edit that data in a way that is appropriate to the application.

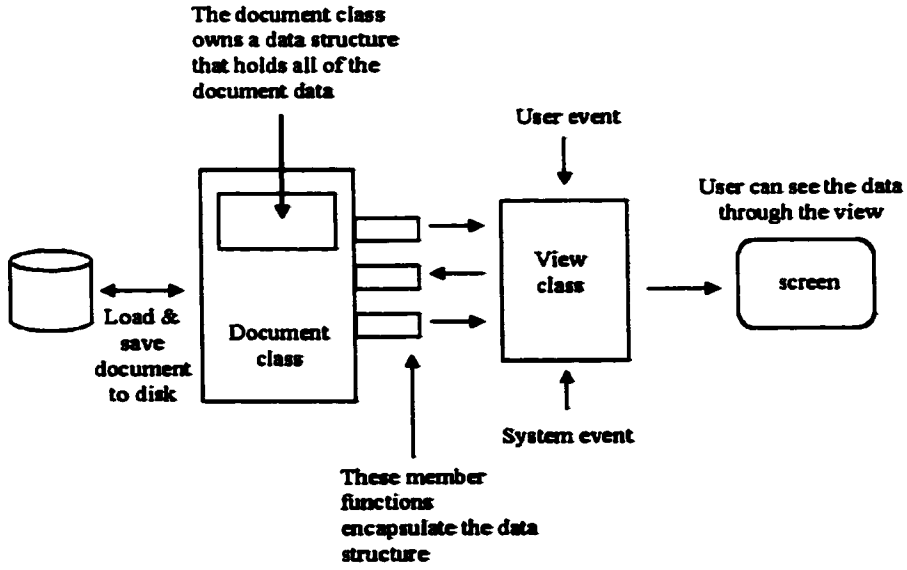

Figure 5.1: The view-document paradigm

The goal of the document class is to completely encapsulate the data for one open document. It holds the data for the document in a data structure in memory and knows how to load the data from the disk and save it to the disk. A view is attached to a document and acts as an intermediary between the document and the user: the view renders an image of the document on the screen and interprets user input as operations upon the document.

An application that has only a single document open at any time is called SDI (Single Document Interface) application. MDI (Multiple Document Interface) application can have multiple documents open at a time. For MDI application, the opened docu-

ments may have several different types, for example, text file or graphic file. Each type of document can have a unique viewing window. Any document may have multiple views open at once. The multiple views for a single document synchronize themselves through the data in the document.

5.2 Architecture of the Program

The program is an MDI Windows application. It provides a graphic user interface in which 3 kinds of documents can be manipulated: OpenGL model, image file and text file. The OpenGL model is used to display the 3D virtual object. The user can pick points from the model or drag the model to watch it from different viewpoints. The image files are displayed in the user interface for the purpose of the calibration or the registration operation; the intermediate result data or final result data can be generated in a text file for future reference, for example, the projection matrix obtained from camera calibration.

The program consists of two independent tools, the semi automatic calibration tool and registration tool. The calibration tool takes two input files: one is the image of calibration pattern grabbed by the camera and the other is a data file that contains the 3D coordinates of the feature points on the calibration pattern. The registration tool takes 5 input files: the OpenGL model file for the virtual object, a pair of stereo images, and 2 calibration data files containing the projection matrices for two cameras that grab the stereo images.

Figure 5.2 shows the MDI architecture of the program. In this architecture, an object of CCVKitApp class (inherited from CWinApp class) is the entry point to the application. Upon instantiating, the CCVKitApp creates an object of CMainFrm class that will hold all child frames and views. Meanwhile, objects of CMultiDocTemplate class are also created for different document types. The document template handles the creation of the document, the frame window and the views. It manages documents of one particular type. In our case, these document types are image document, text document and OpenGL model document.

For an image document, the document template collects 4 items with it:

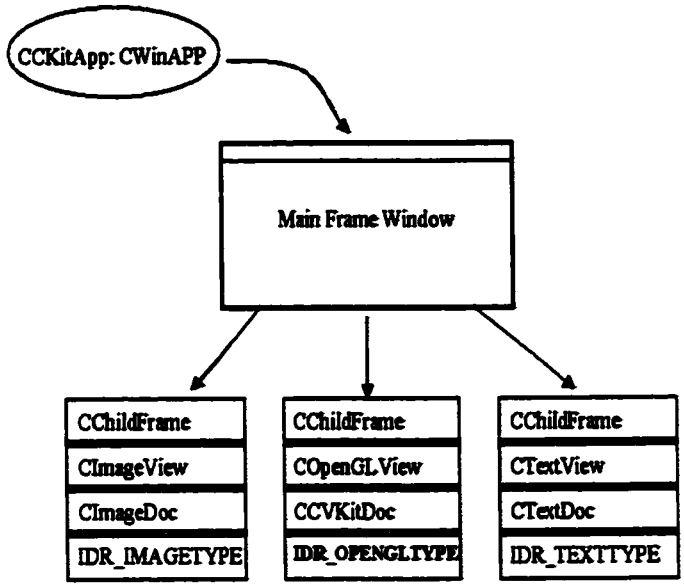

Figure 5.2: The program architecture

- A resource ID identifying a menu resource: IDR_OPENGLTYPE
- A document class: CImageDoc
- A frame window class: CChildFrame
- A view class: CImageView

Similarly, for a text document, the document template collects a resource named IDR_TEXTTYPE, and class CTextDoc, CChildFrame and CTextView with it. For an OpenGL model document, the document template collects IDR_OPENGLTYPE, and class CCVKitDoc, CChildFrame and COpenGLView with it.

The class CCVKitApp will catch all application level messages. They are messages from the menus and the tool bar. When the user selects the menu item for calibration or registration, the message will be caught by the class CCVKitApp. The appropriate dialog box pops up and asks user to select input file. Once the input files are selected, the corresponding views and documents are opened in the child frame for calibration or registration operation.

The CMainFrm class will catch all tools level messages, for example, the messages from buttons associated with calibration tool or registration tool. In figure 5.4, when calibration button is pressed, the message handler function for calibration button in class CMainFrm will call the calibration routine and generates the result text file in child frame.

There are other auxiliary classes, such as classes for computing of homography, geometry, and SUSAN corner detector etc.

5.3 Experiment Results

5.3.1 Camera Calibration

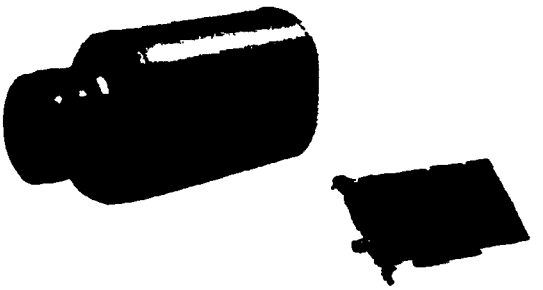

Figure 5.3: Experiment equipment

Figure 5.3 shows the equipment we used in camera calibration experiment. The camera model is COHU 4810, a solid-state CCD camera. The video captured by the camera is input to a frame grabber card, Matrox Meteor-II. The calibration pattern we used is shown in figure 5.4, which is composed of two planes perpendicular to each other. The feature points are the corners of the black squares on the pattern plane. Each pattern plane has 12 squares, therefore the number of feature points is 96.

Figure 5.4 is the screen snapshot of the calibration interface. The crosses marked on the image are the corners detected by the corner detector. Note that not all corners

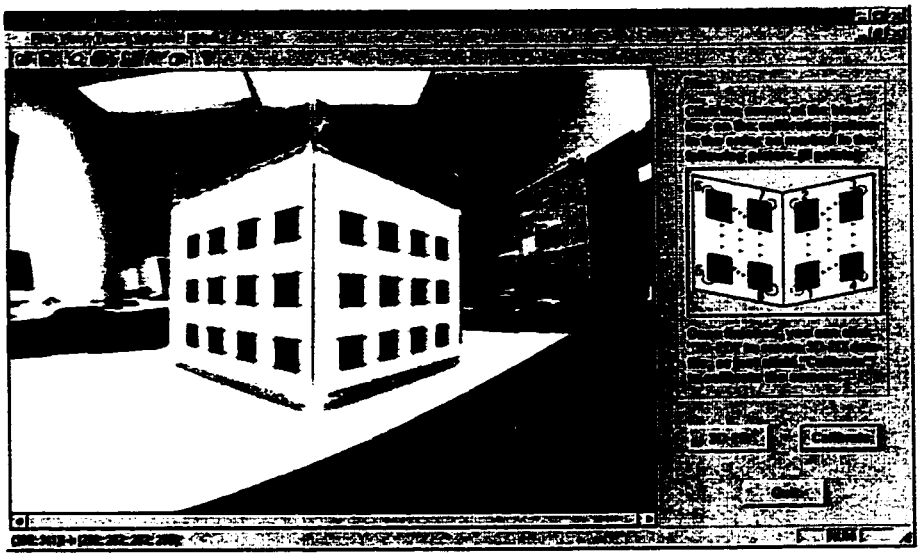

Figure 5.4: Screen snapshot of calibration tool

are detected. Only the detected feature points will be used for calibration. Table 5.1 contains some of the data sheet generated by the corner detector and the homograhy. The correspondences between feature points and their projections have been built.

When the user clicks on the calibration button in the interface, the program performs the calibration using the data correspondences. Table 5.2 contains the calibration result, the projection matrix. Table 5.3 shows the errors between the re-projected points and pixel points. The average error is round 2 pixels.

space coordinates			pixel coordinates	
x	y	z	u	v
39.000000	35.000000	0.000000	326	321
39.000000	65.000000	0.000000	346	317
39.000000	94.000000	0.000000	364	311
39.000000	124.000000	0.000000	379	308
39.000000	154.000000	0.000000	398	303
69.000000	35.000000	0.000000	324	290
69.000000	65.000000	0.000000	346	288
69.000000	94.000000	0.000000	365	285
69.000000	124.000000	0.000000	379	281
69.000000	154.000000	0.000000	396	281
69.000000	184.000000	0.000000	407	279
98.500000	35.000000	0.000000	326	254
98.500000	65.000000	0.000000	345	252
98.500000	94.000000	0.000000	367	253
98.500000	124.000000	0.000000	380	251
98.500000	154.000000	0.000000	398	252

Table 5.1: 3D-2D correspondences built by homography and corner detection

m_{ij}	1	2	3	4
1	-58.745068	405.281403	-53.847713	94815.546875
2	-337.074310	101.869659	101.320351	132233.140625
3	-0.202485	0.712808	0.671495	350.629059

Table 5.2: The projection matrix

pixel coordinates		re-projected pixel coordinates		error
u	v	x	y	$\sqrt{(u-x)^2+(v-y)^2}$
291.000000	331.000000	290.223083	333.585028	2.699254
305.000000	322.000000	305.521942	323.105101	1.222159
318.000000	313.000000	318.792877	314.014318	1.287437
331.000000	305.000000	331.182251	305.527442	0.558041
344.000000	296.000000	342.400604	297.842703	2.440004
351.000000	291.000000	352.606537	290.851511	1.613385
291.000000	313.000000	290.224792	311.224079	1.937742
319.000000	297.000000	319.224365	293.688541	3.319051
331.000000	289.000000	331.769073	286.102973	2.997372
354.000000	275.000000	353.417908	273.012298	2.071181
291.000000	287.000000	290.226532	288.490909	1.679602
305.000000	279.000000	306.014099	280.233807	1.597084
319.000000	272.000000	319.661499	273.096027	1.280178
332.000000	265.000000	332.362701	266.453122	1.497703
344.000000	259.000000	343.830627	260.455245	1.465068

Table 5.3: Calibration errors

5.3.2 3D Registration

Figure 5.5 shows the user interface for object registration. Figure 5.6 shows the matching result when the user performs the reconstruction of the registration plane. Crosses on the first image are the points selected by the user, and they are automatically matched on the second image. The lines drawn on figure 5.6 are the epipolar lines. Table 5.4 shows the coordinates of the reconstructed points that define the registration plane. Figure 5.7 and figure 5.8 shows the registration results.

reconstructed coordinates			real coordinates			error
x	y	z	x'	y'	z'	shift(in mm)
188.549683	1.225690	19.958725	188.500000	0.00000	17.00000	3.20294
38.243710	3.188082	19.118397	39.000000	0.00000	17.00000	3.90172
39.065563	1.888066	139.176773	39.000000	0.00000	136.500000	3.27631

Table 5.4: Reconstruction errors

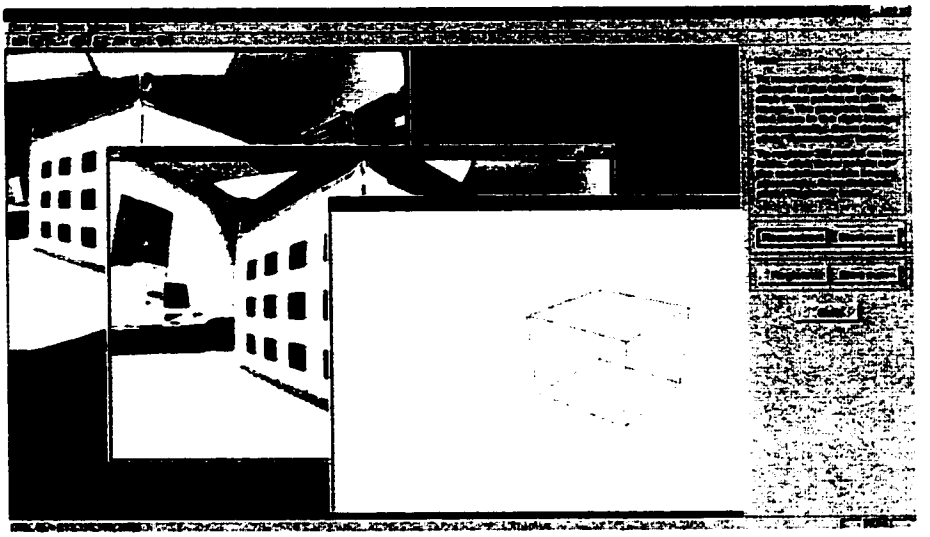

Figure 5.5: Screen snapshot of registration tool

5.4 Conclusion

Our registration program for augmented reality is a Windows application developed with Visual C++ and MFC. In this chapter, we have discussed the architecture of the program from the viewpoint of Windows programming. The program has been thoroughly tested by a large number of experiments. The camera calibration tool is user-friendly. Experiments show that the calibration accuracy is practical enough for the registration. The registration tool has been also made user-friendly. The user can perform a fast and accurate registration with only 6 mouse clicks.

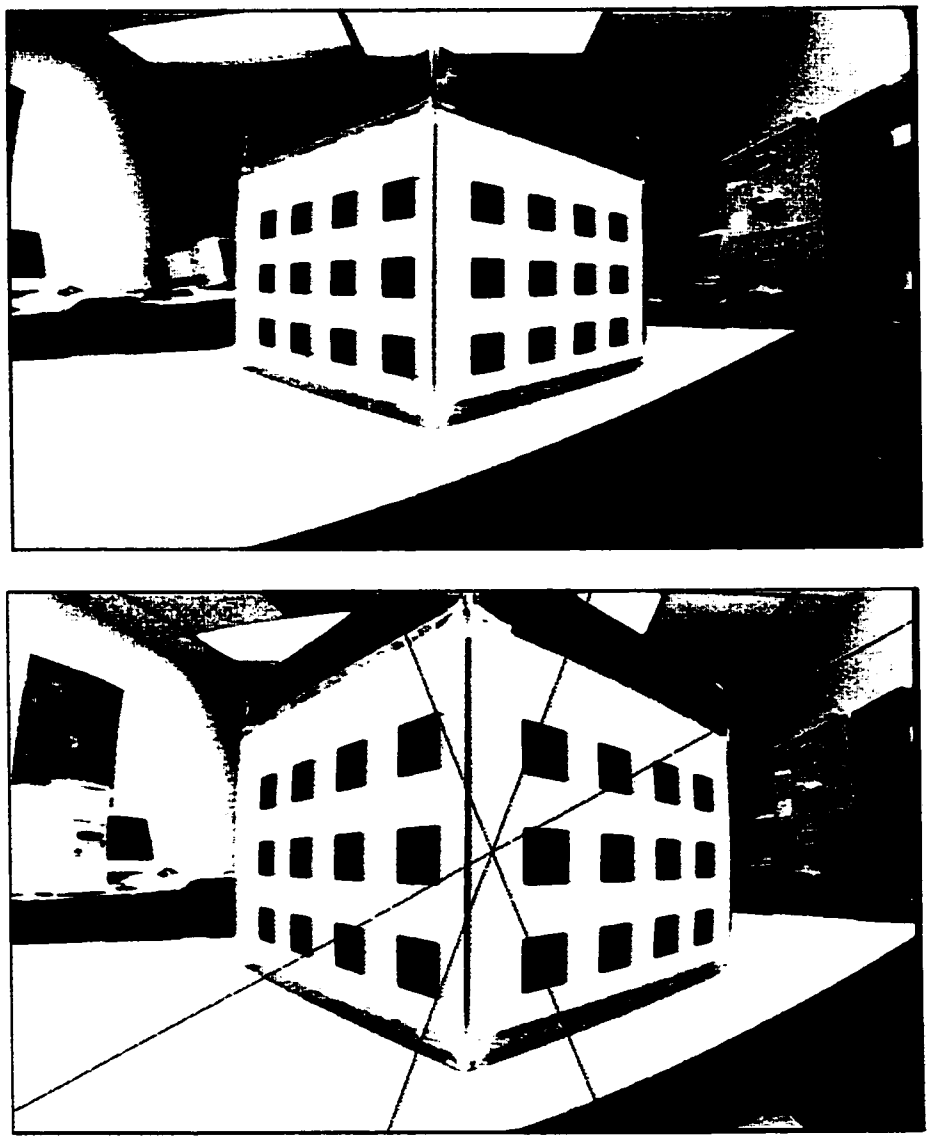

Figure 5.6: Matching results

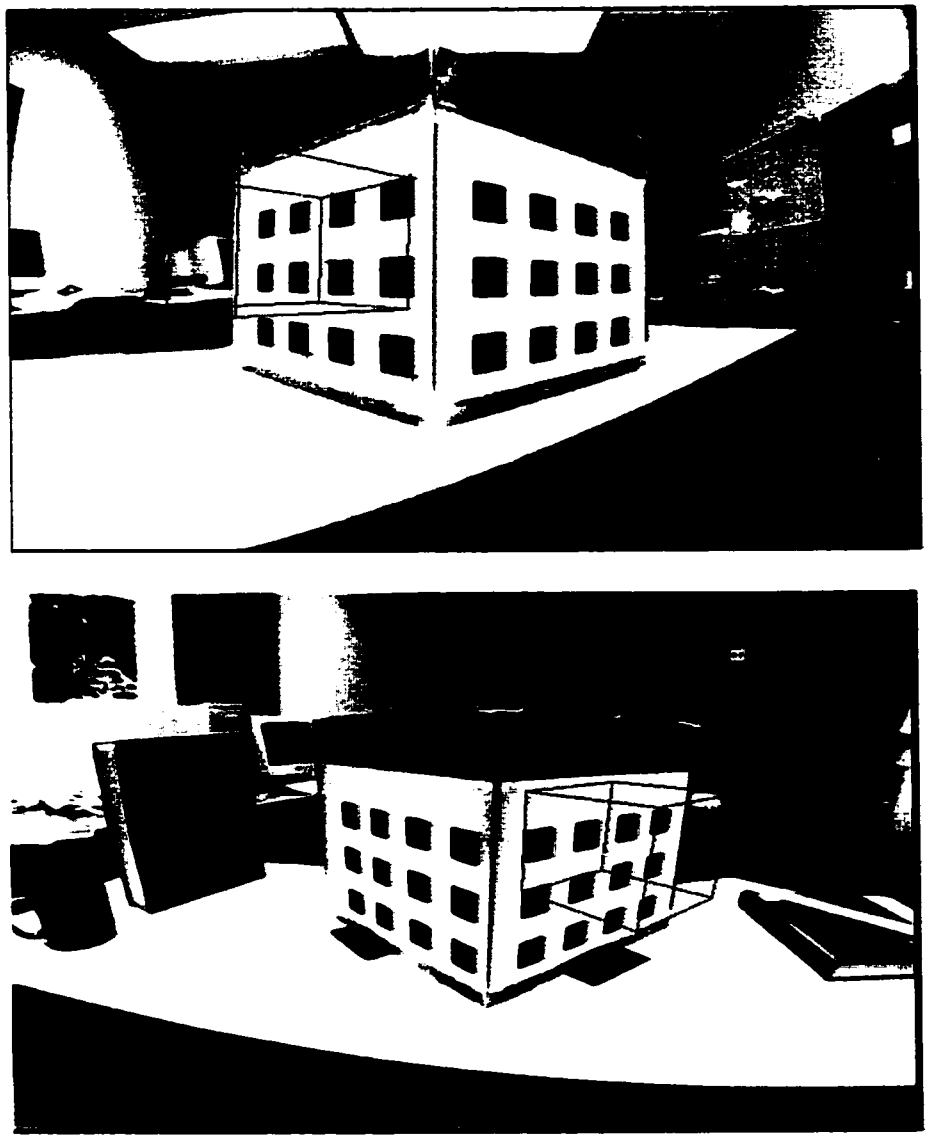

Figure 5.7: Registering a cube on the pattern plane

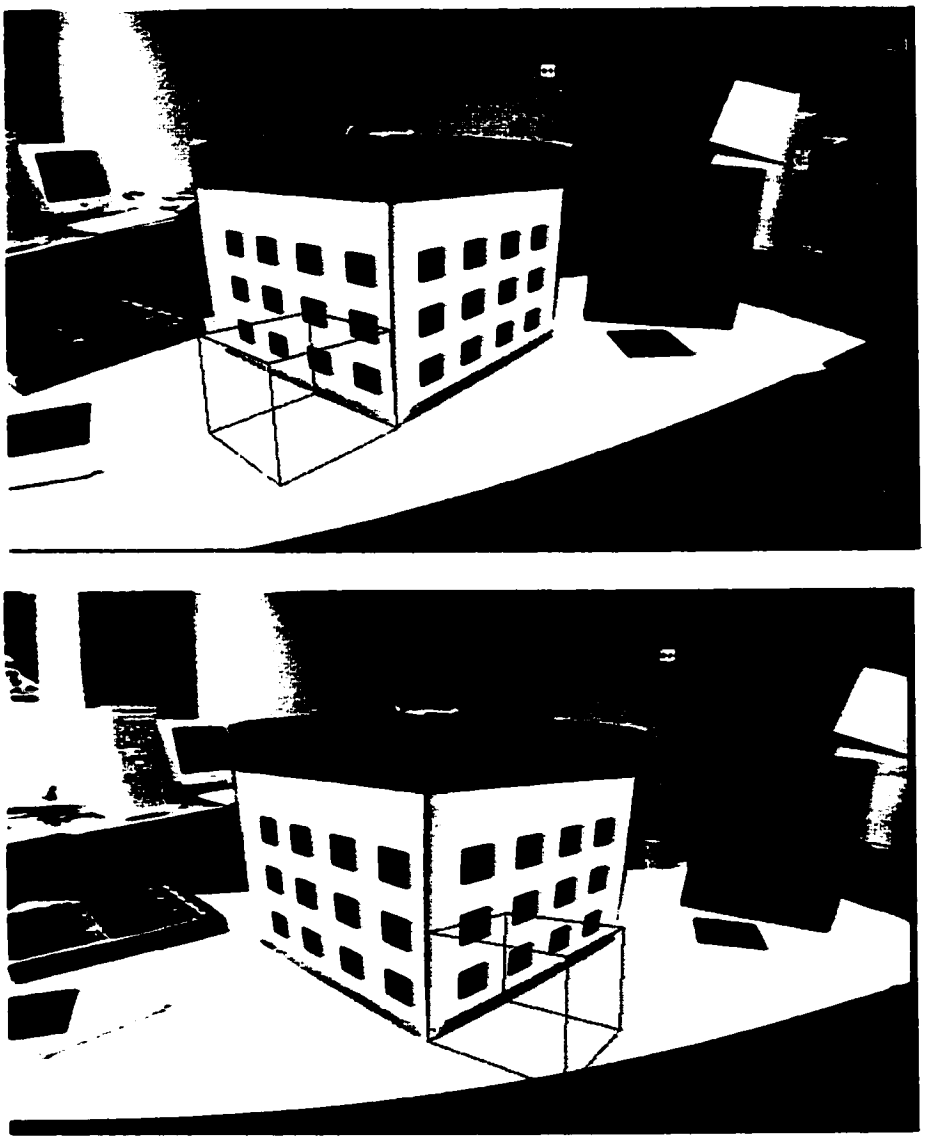

Figure 5.8: Registering a cube on the table

Chapter 6

Conclusion and Future Work

The main contribution of this thesis is the design and implementation of an interactive object registration approach for augmented reality.

The registration problem is a fundamental problem in augmented reality. Without an accurate and practical registration, the AR system becomes meaningless. A good registration will require 3 consistencies:

- Geometrical consistency. When a virtual object is registered in a real environment, it should appear in proper size with right position and orientation with respect to the real environment.
- Motion consistency. The virtual object should act as its real counterpart when the real environment or the viewpoint changes.
- Photometry consistency. The virtual object should produce proper light and shade effects like real object.

In chapter 4 the existing registration techniques are reviewed . The advantage of directly specifying the object's coordinates is the accuracy. However, this method does not enable the user to interact with the system directly. The use of manipulators gives the user the powers to interactively register virtual objects onto a real scene. In order to give the depth information, the manipulator system usually provides a stereo view of the real scene and the user manipulates the virtual object in two views simultaneously.

In theory, manipulators can position the virtual object freely in the real scene. But in practice, the user must be highly trained to accommodate this kind operation. Manipulator can not provide high registration accuracy. Some researchers proposed to put a real object on the desired position and manipulate the virtual object to coincide with the real one so that the registration accuracy will be improved. The stereo position also provides an interactive registration scheme. Compared to the former two methods, it is more practical. However, the user is asked to apply kinds of geometry constraints, for example epipolar constraint, when assigning the projections on stereo view. Thus, it is awkward for users without professional background.

Our method described in this thesis provides the simplest way to register virtual objects in the real scene. Any common user can perform a registration within only 6 mouse clicks. This is because our method integrates some computer vision techniques and makes it highly automatic. The similarity matching, epipolar geometry and 3D reconstruction techniques enable the user to click 3 points on the image to define a plane and click 3 points to register the virtual object. Our method has been thoroughly tested. The experiment results show its capability and efficiency.

We should note that the registration accuracy of our method depends on the accuracy of the camera calibration. The latter's accuracy depends on the accuracy of corner detection and the camera model we use.

Future work

Our method can be improved and extended in following areas:

- Use of corner detector that has sub-pixel accuracy. Our method currently use SUSAN corner detector which provides pixel accuracy. The calibration accuracy based on this corner detector gives an error of about 2 pixels. Though it is practical for our registration, our future plan is to use a corner detector with better accuracy.

- Use of trifocal tensor technique. Instead of using camera pose estimation to

estimate the projection of the virtual object in the image sequence, we will use the tensor to synthesize the virtual object in the remaining images of the sequence.

Bibliography

[1] Marc-Andr Ameller, Bill Triggs, and Long Quan. Camera pose revisited - New linear algorithms, working paper(not accepted for ECCV'00). `http://www.inrialpes.fr/movi/people/Triggs/`, 2000.

[2] Adnan Ansar and Konstantinos Daniilidis. Linear augmented reality registration. In *IEEE Proceedings of Computer Analysis of Images and Patterns*, pages 383–390, 2001.

[3] P. Aschwanden and W. Guggenbuhl. Experimental results from a comparative study on correlation-type registration algorithms. *Robust Computer Vision*, pages 268–282, 1992.

[4] Ronald Azuma. A survey of augmented reality. *Teleoperators and Virtual Environments*, pages 355–385, August 1997.

[5] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, pages 34–47, Nov/Dec 2001.

[6] Ronald Azuma and G. Bishop. Improving static and dynamic registration in an optical See-Through HMD. In *Proceedings of ACM Siggraph 94, ACM*, pages 197–204, New York, 1994.

[7] Ronald Azuma, Howard Neely III, Michael Daily, and Ryan Geiss. Visualization tools for free flight air-traffic management. *IEEE Computer Graphics and Applications*, pages 32–36, Sept/Oct 2000.

[8] Ronald Azuma, Jong Weon Lee, Bolan Jiang, Jun Park, Suya You, and Ulrich Neumann. Tracking in unprepared environments for augmented reality systems. *Computers and Graphics*, pages 787–793, December 1999.

[9] Ronald Azuma, Yuichi Ohta, and Hideyuki Tamura. *The Challenge of Making Augmented Reality Work Outdoors, In Mixed Reality: Merging Real and Virtual Worlds*, chapter 21, pages 379–390. Springer-Verlag, ISBN 3-540-65623-5, 1999.

[10] Ronald Tadao Azuma. *Predictive Tracking for Augmented Reality.* PhD thesis, University of North Carolina, February 1995.

[11] Michael Bajura. Merging real and virtual environments with video See-Through Head-Mounted displays. Technical Report TR98-036, University of North Carolina, 1998.

[12] Michael Bajura and Ulrich Neumann. Dynamic registration correction in video-Based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–61, 1995.

[13] S. Baker, F. Dellaert, and I. Matthews. Aligning images incrementally backwards. Technical Report CMU-RI-TR-01-03, CMU Rob. Institute, 2001.

[14] M. Berger. Resolving occlusions in augmented reality: a contour-based approach without 3D reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, 1997.

[15] Marie-Odile Berger and Gilles Simon. Robust image composition algorithms for augmented reality. In *IEEE Proceedings of Asia Conference on Computer Vision (2)*, pages 360–367, 1998.

[16] Boeing. Boeing technique report. `http://esto.sysplan.com/ESTO/Displays/HMDTDS/Factsheets/Boeing.html`, July 1994.

[17] 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.

[18] David E. Breen, Ross T. Whitaker, Eric Rose, and Mihran Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22, 1996.

[19] Rodrigo L. Carceroni and Christopher M. Brown. Numerical methods for model-based pose recovery. Technical Report TR659, Computer Science Department University of Rochester, 1997.

[20] Chu-Song Chen, Yi-Ping Hungand, Sheng-Wen Shih, Chen-Chiung Hsieh, Cheng-Yuan Tang, Chih-Guo Yu, and You-Chung Cheng. Integrating virtual objects into real images for augmented reality. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology, VRST/98*, November 1998.

[21] Li-Te Cheng and John Robinson. Dealing with speed and robustness issues for video-based registration on a wearable computing platform. In *IEEE Proceedings of International Symposium on Wearable Computers*, pages 84–91, 1998.

[22] Stephane Christy and Radu Horaud. Fast and reliable object pose estimation from line correspondences. In *IEEE Proceedings of Computer Analysis of Images and Patterns*, pages 432–439, 1997.

[23] D. Chung, I. Yun, and S. Lee. Registration of multiple range views using the reverse calibration technique. *IEEE Transactions on Pattern Recognition*, 31(4), Apr 1998.

[24] Fabio Gagliardi Cozman, Eric Krotkov, and Carlos Guestrin. Outdoor visual position estimation for planetary rovers. *Autonomous Robots*, 9(2):135–150, 2000.

[25] L. de Ipina. TRIP: A distributed vision-based sensor system. Technical Report PhD 1st Year Report, University of Cambridge, Cambridge, UK, 1999.

[26] Daniel DeMenthon and Larry S. Davis. Model-based object pose in 25 lines of code. In *IEEE Proceedings of European Conference on Computer Vision*, pages 335–343, 1992.

[27] Mihran Tuceryan Department. Single point active alignment method (SPAAM) for optical see-through HMD calibration for AR. In *IEEE Proceedings of International Symposium on Augmented Reality*, 2000.

[28] Chitra Dorai, Juyang Weng, and Anil K. Jain. Optimal registration of object views using range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1131–1138, 1997.

[29] George Drettakis, Luc Robert, and Sylvain Bougnoux. Interactive common illumination for computer augmented reality. In Julie Dorsey and Phillipp Slusallek, editors, *IEEE Proceedings of Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 45–56, New York, NY, 1997. Springer Wien.

[30] Steven Feiner, Blair MacIntyre, and Doree Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):52–62, July 1993.

[31] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, pages 24:381–395, 1981.

[32] Smith Fitzgibbon. Improving augmented reality using image and scene constraints. In *IEEE Proceedings of The British Machine Vision Conference*, 1999.

[33] A.L. Fuhrmann, R. Splechtna, and J. Pikryl. Comprehensive calibration and registration procedures for augmented reality (working paper). `http://www.vrvis.at/research/publications.html`, Oct 2001.

[34] Anton Fuhrmann, Gerd Hesina, François Faure, and Michael Gervautz. Occlusion in collaborative augmented environments. *Computers and Graphics*, 23(6):809–819, 1999.

[35] S. Gottschalk and J. Hughes. Autocalibration for virtual environments tracking hardware. *Computer Graphics*, pages 65–72, 1993.

[36] W. E. L. Grimson, T. Lozano-Pérez, 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 *IEEE Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, pages 430–436, Seattle WA, June 1994.

[37] Carceroni Harman. *The Confluence of Vision and Control (Real-Time Pose Estimation and Control for Convoying Applications)*, pages 230–243. Springer-Verlag,, 1998.

[38] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of CVPR97*, 1997.

[39] Axel Hildebrand. Augmented reality, What for? `http://www.inigraphics.net/publications/topics/1998/issue1/1_98Artikel.%pdf`.

[40] Tobias H?llerer, Steven Feiner, Tachio Terauchi, Gus Rashid, and Drexel Hallaway. Exploring MARS: Developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers and Graphics*, 23(6):779–785, 1999.

[41] Bruce Hoff and Ronald Azuma. Autocalibration of an electronic compass in an outdoor augmented reality system. *Proceedings of International Symposium on Augmented Reality*, pages 159–164, October 2000.

[42] William A. Hoff and Khoi Nguyen. computer vision-based registration techniques for augmented reality. In *SPIE Proceedings of Intelligent robots and computer vision XV*, volume 2094 18-22, pages 538–548, boston, MA, 1996.

[43] T. Hollerer, S. Feiner, D. Hallaway, B. Bell, S. Julier, Y. Baillot, D. Brown, and M. Lanzagorta. User interface management techniques for collaborative mobile augmented reality. In *Proceedings of Computer and Graphics*, Elsevier, 2001.

[44] H. Holloway. Registration error analysis for augmented reality. *Teleoperators and Virtual Environments*, 6(4):413–432, August 1997.

[45] Richard Lee Holloway. Registration errors in augmented reality systems. Technical Report TR95-016, University of North Carolina, `http://www.cs.unc.edu/Research/techreports.html`, Jan 1995.

[46] R. Horaud, S. Christy, and F. Dornaika. Object pose: the link between weak perspective, paraperspective, and full perspective (working paper). `http://citeseer.nj.nec.com/horaud97object.html`, 1994.

[47] Jun'ichi Hoshino, Hirofumi Saito, and Masanobu Yamamoto. Automatic registration of virtual objects onto human image sequences. In *IEEE Proceedings of the International Conference on Pattern Recognition (ICPR'00)*, 2000.

[48] Marco C. Jacobs, Mark A. Livingston, and Andrei State. Managing latency in complex augmented reality systems. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 49–54, 185, 1997.

[49] Bolan Jiang, Suya You, and Ulrich Neumann. Camera tracking for augmented reality media. In *Proceedings of IEEE International Conference on Multimedia and Expo (III)*, pages 1637–1640, 2000.

[50] Yi Murphey Jianxin. Registering real-scene to virtual imagery using robust image features. In *Proceedings of Vision, Modelling and Visualization (VMV01)*, Stuttgart Germany, November, 2001.

[51] S. Julier, Y. Baillot, D. Brown, M. Lanzagorta, and L. Rosenblum. Bars: The battlefield augmented reality system. In *Proceedings of the NATO Information Systems Technology Panel Symposium on New Information Processing Techniques for Military Systems*, 2000.

[52] S. Julier, M. Lanzagorta, Y. Baillot, L. Rosenblum, T. Hollerer, S. Feiner, and S. Sestito. Information filtering for mobile augmented reality. In *Proceedings of the IEEE International Symposium on Augmented Reality*, 2000.

[53] M. Kanbara, H. Fujii, H. Takemura, and N. Yokoya. A stereo vision-based augmented reality system with an inertial sensor. In *IEEE Proceedings of ISAR*, 2001.

[54] Lipson Shpitalni Kimura. Online product maintenance by web-based augmented reality (working paper). `www.mae.cornell.edu/lipson/papers/augmaint.pdf`.

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

[56] K. Kutulakos and J. Vallino. Affine object representations for calibration-free augmented reality. In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 1996.

[57] Simon Lepetit. Computer vision methods for registration: Mixing 3D knowledge and 2D correspondences for accurate image composition (working paper). `citeseer.nj.nec.com/129137.html`.

[58] Mark Alan Livingston. *Vision-based Tracking with Dynamic Structured Light for Video See-through Augmented Reality.* PhD thesis, University of North Carolina, February 1995.

[59] Chien-Ping Lu, Gregory D. Hager, and Eric Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000.

[60] W. Luk, T. K. Lee, J. R. Rice, N. Shirazi, and P. Y. K. Cheung. Reconfigurable computing for augmented reality. In Kenneth L. Pocek and Jeffrey Arnold, editors, *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pages 136–145, Los Alamitos, CA, 1999. IEEE Computer Society Press.

[61] B. MacIntyre and E. Coelho. Adapting to dynamic registration errors using level of error. In *IEEE Proceedings of International Symposium on Augmented Reality'2000*, pages 85–88, Munich, Germany, October 2000.

[62] Pattie Maes. Artificial life meets entertainment: Lifelike autonomous agents. *Communications of the ACM*, 38(11):108–114, November 1995.

[63] J. B. Antoine Maintz and Max A. Viergever. An overview of medical image registration methods. Technical report, Universiteit Utrecht, 1998.

[64] E. McGarrity and M. Tuceryan. A method for calibrating see-through head-mounted displays for augmented reality. Technical Report TR–CIS–0499-13, University Purdue University Indianapolis, Indiana, 1999.

[65] Yakup Genc Mihran. Optical See-Through calibration with Vision-Based trackers: Propagation of projection matrices. In *Proceedings of International Symposium on Augmented Reality)*, 2001.

[66] P. Milgram and D. Drascic. Perceptual effects in aligning virtual and real objects in augmented reality displays (working paper). `http://citeseer.nj.nec.com/423465.html`, 1997.

[67] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12):1321C1329, 1994.

[68] Li Ming. Correspondence analysis between the image formation pipelines of graphics and vision. In *Proceedings of IX Spanish Symposium on Pattern Recognition and Image Processing*, May 2001.

[69] U. Neumann and Y. Cho. A Self-Tracking augmented reality system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, July 1996.

[70] Ulrich Neumann, Suya You, Youngkwan Cho, Jongweon Lee, and Jun Park. Augmented reality tracking in natural environments(working paper). `http://citeseer.nj.nec.com/479903.html`.

[71] Toshikazu Ohshima, Kiyohide Satoh, and et al. Ar2hockey: A case study of collaborative augmented reality. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS'98)*, Atlanta USA, 1998.

[72] Siu-Hang Or, W. S. Luk, K. H. Wong, and Irwin King. An efficient iterative pose estimation algorithm. In *IEEE Proceedings of ACCV (2)*, pages 559–566, 1998.

[73] J. Park, B. Jiang, and U. Neumann. Vision-based pose computation: Robust and accurate augmented reality tracking. In *Proceedings of IEEE International Workshop on Augmented Reality*, pages 3–12, 1999.

[74] Long Quan and Zhong-Dan Lan. Linear N-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, 1999.

[75] R. Raskar, G. Welch, and W. Chen. Table-top spatially augmented reality: Bringing physical models to life using projected imagery. In *IEEE Proceedings of Second Int Workshop on Augmented Reality (IWAR'99)*, San Francisco, CA, 1999.

[76] S. Ravela, B. Draper, J. Lim, and R. Weiss. Tracking object motion across aspect changes for augmented reality. *IUW*, pages 1345–1352, 1996.

[77] S. Ravela, T. Schnackertz, R. Grupen, R. Weiss, E. Riseman, and A. Hanson. Temporal registration for assembly. In *IEEE proceedings of IROS Computer Vision Workshop*, 1995.

[78] Michael K. Reed and Peter K. Allen. 3-D modeling from range imagery: An incremental method with a planning component. *Image and Vision Computing 17*, pages 99–111, 1999.

[79] B. Reinhold. Improving the registration precision by visual horizon silhouette matching. In *IEEE Proceedings of the First IEEE Workshop on Augmented Reality*, San Francisco, CA., November 1998.

[80] G. Reitmayr and D. Schmalstieg. Mobile collaborative augmented reality. In *IEEE Proceedings of ISAR*, 2000.

[81] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings of the Asia Pacific Computer Human Interaction Conference (APCHI'98), Japan*, July 1998.

[82] M. Ribo, A. Pinz, and A. Fuhrmann. A new optical tracking system for virtual and augmented reality applications. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference, IMTC*, 2001.

[83] E. Rose, D. Breen, K Ahlers, D. Greer, C. Crampton, M. Tuceryan, and R. Whitaker. Annotating real-world objects using augmented vision. In *IEEE Proceedings of Computer Graphics International '95*, 1995.

[84] Bodo Rosenhahn, Yiwen Zhang, and Gerald Sommer. Performance of constraint based pose estimation algorithms (working paper). `http://citeseer.nj.nec.com/373727.html`.

[85] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):1–12, 1999.

[86] Christian Schtz and Heinz Hgli. Augmented reality using range images. In *Proceedings of SPIE Photonics West, The Engineering Reality of Virtual Reality*, 1997.

[87] Miguel Ribo September. State of the art report on optical tracking. Technical report, University of North Carolina, 1998.

[88] Gilles Simon and Marie-Odile Berger. A two-stage robust statistical method for temporal registration from features of various type. Technical Report RR-3235, INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 1998.

[89] Andrei State, Gentaro Hirota, David T. Chen, William F. Garrett, and Mark A. Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. *Computer Graphics*, 30(Annual Conference Series):429–438, 1996.

[90] Blaine Bell Steven. View management for virtual and augmented reality. In *Proceedings of ACM Symp. on user interface software and technology*, pages 101–110, 2001.

[91] Tobias Hollerer Steven. User interface management techniques for collaborative mobile augmented reality. *Computers and Graphics*, 25(5):799–810, 2001.

[92] Peter Sturm. Algorithms for plane-based pose estimation. In *IEEE Proceedings of the Computer Vision and Pattern Recognition*, 2000.

[93] R. Torre, S. Balcisoy, P. Fua, and D. Thalmann. Interaction between real and virtual humans: Playing checkers. In *IEEE Proceedings of Eurographics Workshop On Virtual Environments*, 2000.

[94] W. Triggs. Camera pose and calibration from 4 or 5 known 3D points. In *Proceedings of IEEE International Conference on Computer Vision*, pages 278–284, 1999.

[95] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, Edinburgh UK, 1998.

[96] R. Y. Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Transaction on Robotics and Automat*, 3(4):323C344, 1987.

[97] Mihran Tuceryan, Douglas S. Greer, Ross T. Whitaker, David E. Breen, Chris Crampton, Eric Rose, and Klaus H. Ahlers. Calibration Requirements and Procedures for a Monitor-Based Augmented Reality System. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):255–273, 1995.

[98] M. Uenohara and T. Kanade. Vision-Based object registration for real-time image overlay. In Nicholas Ayache, editor, *Proceedings of Computer Vision, Virtual Reality and Robotics in Medicine*. Springer-Verlag, 1995.

[99] Alberto Valinetti, Andrea Fusiello, and Vittorio Murino. Model tracking for video-based virtual reality (working paper). `http://citeseer.nj.nec.com/447446.html`.

[100] Ross T. Whitaker, Chris Crampton, David E. Breen, Mihran Tuceryan, and Eric Rose. Object calibration for augmented reality. *Computer Graphics Forum*, 14(3):15–28, 1995.

[101] Fung Wong. A real-time iterative three-point model-based pose-estimation algorithm (working paper). `http://citeseer.nj.nec.com/146164.html`.

[102] You, Suya, Ulrich Neumann, and Ronald Azuma. Hybrid inertial and vision tracking for augmented reality registration. *Proceedings of IEEE VR '99*, pages 260–267, March 1999.

[103] Suya You. Fusion of vision and gyro tracking for robust augmented reality registration. In *IEEE Proceedings of Virtual Reality 2001 Conference (VR'01)*, 2001.

[104] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *IEEE Proceedings of International Conference on Computer Vision (1)*, pages 666–673, 1999.

[105] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

Vita Auctoris

NAME: Zonglei Huang

EDUCATION: University of Windsor, Windsor, ON, Canada
2000-2002 M.Sc.

Zhejiang University, Zhejiang, China
1995-1998 M.Eng.