AUGMENTED REALITY INTERACTION AND VISION-BASED TRACKING

XU KE

NATIONAL UNIVERSITY OF SINGAPORE

2003

AUGMENTED REALITY INTERACTION AND VISION-BASED TRACKING

XU KE

B.Eng.(Hons), NUS

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgement

I wish to thank Dr. Adrian David Cheok and Dr. Simon J.D. Prince for their patient and invaluable guidance.

Contents

Abstract

Augmented Reality (AR) refers to the incorporation of virtual objects into a real three dimensional scene. In this thesis work, tangible user interfaces were developed for various real-time AR applications in different environments. A vision-based tracking technique by using fiducial markers was adopted for accurate registration for desktop AR applications. An interactive AR system for virtual *Hello Kitty Garden* was developed in this thesis work. In this application, the command interface to the computer has been merged with people's everyday interaction with the environment, such as picking or dropping. The interface is thus universal. However, for most outdoor AR applications, the environment is necessarily unprepared, i.e. no fiducial markers were used. To solve this problem, a new type of vision-based tracking technique was proposed here in this thesis work, which makes use of natural features of the environment. Two different types of image motion constraints are studied: the epipolar constraint and the homography. In particular, a homography can exactly describe the image motion when the scene is planar, or when the camera movement is a pure rotation, and provides a good approximation when these conditions are nearly met. The calculation for both of these two image motion constraints is based on stored representations of the scene and prevents a gradual drift in the augmentation position error. A real-time tracking algorithm based on homographies was developed for both indoor and outdoor AR applications. At the end, we assessed all these natural tracking algorithms across a number of criteria including robustness, speed and accuracy.

Three papers based on this thesis work have been published or accepted for international journals and conferences [1] [2] [3].

List of Figures

Chapter 1

Introduction

The current trend towards pervasive computing suggests future work environments will comprise of a range of information displays and interaction devices. These will include normal desktop computers or even notebook computers together with 3D immersive displays. Recently, there have been a lot of research work on the creation of new interaction systems using Augmented Reality (AR).

Augmented Reality (AR) is different from Virtual Reality (VR), however there is no clear borders between these two technologies. VR technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or merged with the real world. Therefore, AR supplements reality, rather than completely replacing it. Ideally, it would appear to the user that the virtual and real objects coexisted in the same space.

Figure 1.1 shows an example of what this might look like. In this example, a user is holding a real book in a real environment. On the book he's holding, there are two virtual cartoon characters (the Hello Kitty and the Kerropi). Note that all the objects are combined in 3-D, so that the virtual Kerropi covers part of the

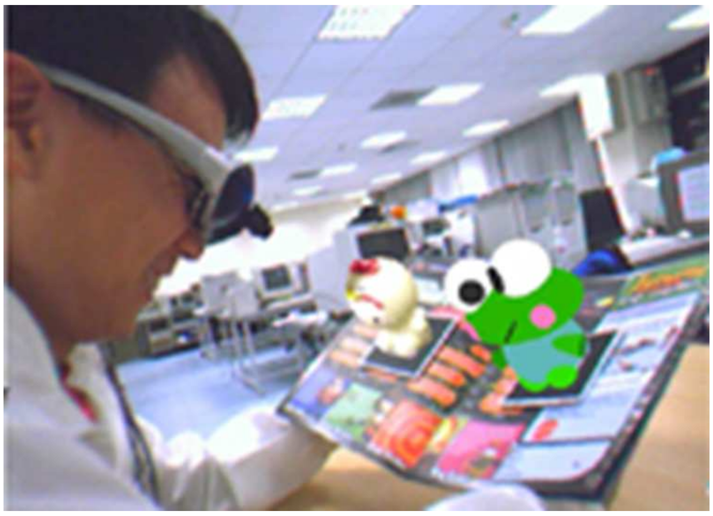

Figure 1.1: A real book with two virtual cartoon characters augmented on the pages.

real book, and appears to be standing on the page. The user will receive the same vision effect for the virtual Hello Kitty on the other page.

1.1 General overview on Augmented Reality

There is no sharp borders between the concepts of Reality, Augmented Reality, and Virtual Reality, instead it can be seen as a continuum spreading from totally real and totally virtual, as proposed by P. Milgram in [4]. The continuum starts at Reality, spreads through Augmented Reality [5] to Virtual Reality as shown in Figure 1.2.

AR is an attractive concept because it can potentially enhance a user's perception of and interaction with the real world [5]. In AR systems, the virtual objects display information that the user cannot directly detect with his own senses. The information conveyed by the virtual objects helps a user perform real-world tasks. AR is a specific example of what Fred Brooks calls Intelligence Amplification (IA): using the computer as a tool to make a task easier for a human to perform [6].

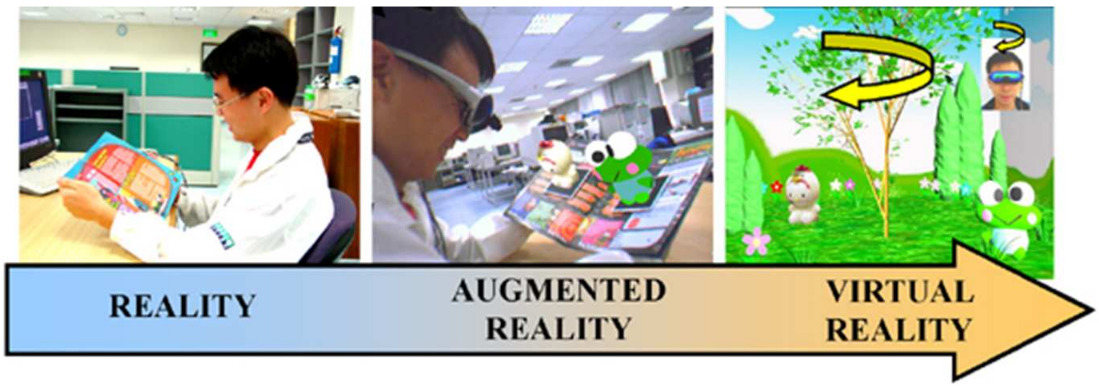

Figure 1.2: Augmented reality (center) can be used as a transitional interface between the real world (left) and virtual environments (right).

In AR applications, the system provides visual aids to the users in realtime, which enhances users ability to accomplish their jobs more efficiently and accurately. Most importantly, AR technology can display those information in front of the users eyes with absolutely no disruption to their ongoing work, because the users still can see the real environment at the same time. More information about development of AR research can be found several papers .

The beginnings of AR date back to Sutherland's work in the 1960s, which used a see-through Head Mounted Display (HMD) to represent 3D graphics. However, in the past few years, many researchers have broaden the definition of AR beyond this vision. According to one of the latest survey in this field [7], the AR system can be defined as a system which has the following properties:

1. combines real and virtual objects in a real environment.

2. runs interactively, and in real time.

3. registers (aligns) virtual objects to physical objects and locations.

Note that we don't restrict this definition of AR to particular display technologies, such as a HMD. Nor do we limit it to our sense of sight. AR can potentially apply to all senses, including hearing, touch, and smell. However, as the vision-based AR technologies have the greatest potential in the new age human-computer interaction applications, we only focus on vision technologies for the purpose of this thesis work.

In vision-based AR systems, a basic design decision is how to accomplish the combining of real and virtual objects. It was usually done by using a see-through HMD. A see-through HMD is one device used to combine real and virtual. It lets the user see the real world, with virtual objects superimposed on it. Two basic choices are available: optical see-through approach, and video see-through approach.

1.1.1 Optical see-through AR Systems

Optical see-through HMDs work by placing optical combiners in front of the user's eyes. These combiners are partially transmissive, so that the user can look directly through them to see the real world. The combiners are also partially reflective, so that the user sees virtual images bounced off the combiners from head mounted monitors. This approach is similar in nature to Head-Up Displays (HUDs) commonly used in military aircraft, except that the combiners are attached to the head. Thus, optical see-through HMDs have sometimes been described as a "HUD on a head". Figure 1.3 shows a conceptual diagram of an optical see-through HMD.

The optical combiners usually reduce the amount of light that the user sees from the real world. Since the combiners act like half-silvered mirrors, they only let in some of the light from the real world, so that they can reflect some of the light from the monitors into the user's eyes. They still can be used as a pair of sunglasses when the supply power to the HMD is been cut off.

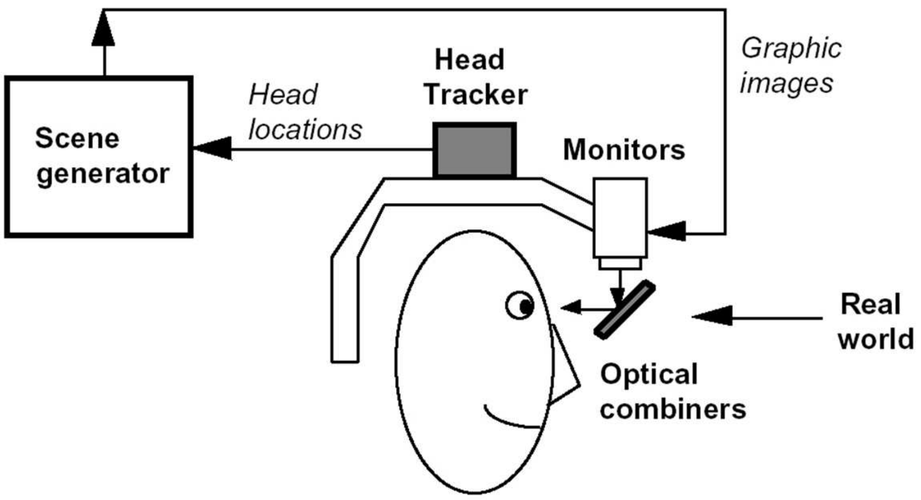

Figure 1.3: Optical see-through HMD conceptual diagram.

1.1.2 Video see-through AR Systems

A basic problem with commercial optical see-through is that the virtual objects do not completely obscure the real world objects, because the optical combiners allow light from both virtual and real sources. Building an optical see-through HMD that can selectively shut out the light from the real world is difficult. In a normal optical system, the objects are designed to be in focus at only one point in the optical path: the user's eye. Any filter that would selectively block out light must be placed in the optical path at a point where the image is in focus, which obviously cannot be the user's eye. Therefore, the optical system must have two places where the image is in focus: at the user's eye and the point of the hypothetical filter. This makes the optical design much more difficult and complex. No existing commercial optical see-through HMD blocks incoming light in this fashion. Thus, the virtual objects appear ghost-like and semi-transparent. This damages the illusion of reality because occlusion is one of the strongest depth cues.

In contrast, video see-through HMDs work by combining a closed-view HMD with one or two head-mounted video cameras. The video cameras provide the user's view of the real world. Video from these cameras is combined with the graphic images created by the scene generator, blending the real and virtual. The result is sent to the monitors in front of the user's eyes in the closed-view HMD. Figure 1.4 shows a conceptual diagram of a video see-through HMD.

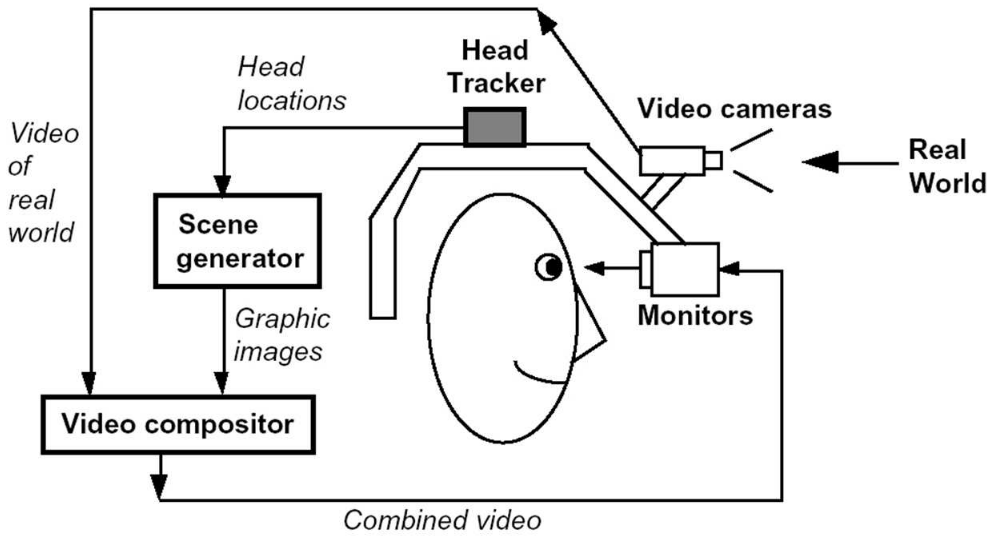

Figure 1.4: Video see-through HMD conceptual diagram.

Compared to optical see-through, video see-through is far more flexible about how it merges the real and virtual images. Since both the real and virtual are available in digital form, video see-through compositors can, on a pixel-by-pixel basis, take the real, or the virtual, or some blend between the two to simulate transparency. Because of this flexibility, video see-through may ultimately produce more compelling environments than optical see-through approaches. However, video see-through HMDs also have their own limitations. Comparing to the optical see-through, they have the inevitable loss of resolution of the physical visual environment. Matching of the field of view of the camera with the field of view of the HMD is another problem for the video see-through. Also, when the resolution of the camera is different from the resolution of the HMD display, we need to match them as well.

Both optical and video technologies have their roles, and the choice of technology depends on the application requirements. As for the purpose of this thesis work, video see-through approach is adopted for its capability of pixel level manipulation.

1.2 Contributions of this Thesis Work

The objective of this thesis work is to develop accurate and robust real time Augmented Reality Systems for both indoor and outdoor applications. The main contributions can be categorized into the following three parts:

1. Based on Mark Billinghurst and Kato's previous work [8], I have developed a Tangible User Interface (TUI) for desktop AR applications. Such interface will allow users to use computer-generated entities (the virtual objects) just as I use physical objects, selecting and manipulating them with our hands instead of with a special-purpose device such as a mouse or joystick. Interaction would then be intuitive and seamless because I would use the same tools to work with digital and real objects.

2. Investigated the current vision-based tracking algorithms, and introduced a new robust and efficient approach to solve the registration problems in unprepared environments by using natural features.

3. Several realtime AR systems based on the proposed algorithm were built for both robust indoor and outdoor applications. As I will discuss in the later chapters, the systems can achieve sub-pixel accuracy in those applications.

The proposed natural feature tracking algorithm is based on always calculating camera pose relative to the pre-captured reference image of the scene. The camera pose of the current image frame relative to the reference image is estimated by matching detected corner points across the image frames and minimizing a cost function based on two-view image constraints. Camera pose estimates of previous image frames provide the starting point for this minimization as well as to regularize the error surface when the incoming data is impoverished.

The two main advantages of the proposed system over previous methods are as follows:

1. The proposed algorithm can be calculated reliably at camera frame rate (about 30 fps) on a normal desktop PC. To the best of knowledge, all of the previous methods based on natural features suffer from the problem of balance between computational load and accuracy.

2. The proposed algorithm is robust and maintains accurate estimates of the camera pose when the incoming images frames matches the minimum requirements for tracking. This robustness is achieved by a temporal regularization technique.

3. This new algorithm has great flexibility. It can be applied to both indoor and outdoor AR applications.

Three papers based on this thesis work have been published or accepted for international journals and conferences [1] [2] [3].

- Published as full paper for the *International Symposium on Wearable Computers* (ISWC), Seattle, Washington, 2002.

- Published as full paper for the *IEEE Transactions on Computer Graphics and Applications*, 2002.

- Published as full paper for the *Journal on Personal and Ubiquitous Computing*, Springer-Verlag London, 2003.

1.3 Organization of Chapters

The structure of this thesis report is as follows: Chapter 2 gives a background overview of the Augmented Reality system, with the focus on vision-based tracking system. Chapter 3 introduces a tangible desktop AR interface, where the interactions between user and virtual objects become intuitive and seamless. A new robust realtime natural feature tracking algorithm for AR systems in unprepared environments is described in Chapter 4. Chapter 5 presents some applications developed based on this new algorithm. A detailed system performance and assessment is shown in Chapter 6.

Chapter 2

Background of Augmented Reality Research

2.1 Main Approaches for the Research

The beginnings of AR, as we define it, date back to the 1960s. However, only over the past decade has there been enough work to refer to AR as a research field. In 1997, Azuma published a survey [5] that defined the field, described many problems, and summarized the developments up to that point. Since then, AR's growth and progress have been remarkable.

One of the most basic problems currently limiting Augmented Reality applications is the registration problem. The objects in the real and virtual worlds must be properly aligned with respect to each other, or the illusion that the two worlds coexist will be compromised.

Registration problems also exist in Virtual Environments, but they are not nearly as serious because they are harder to detect than in Augmented Reality. Since the user only sees virtual objects in VE applications, registration errors result in visual-kinesthetic and visual-proprioceptive conflicts. Because the kinesthetic and proprioceptive systems are much less sensitive than the visual system, visual-kinesthetic and visual-proprioceptive conflicts are less noticeable than visual-visual conflicts. For example, a user wearing a closed-view HMD might hold up her real hand and see a virtual hand. This virtual hand should be displayed exactly where she would see her real hand, if she were not wearing an HMD. But if the virtual hand is wrong by five millimeters, she may not detect that unless actively looking for such errors. The same error is much more obvious in a see-through HMD, where the conflict is visual-visual.

There are basically two different approaches to achieve accurate registration and positioning of virtual objects in the real environment: the sensing by long-range hardware trackers, or using various vision-based methods.

2.1.1 Hardware Sensing and Tracking Methods

Sensors, based on magnetic, mechanical, ultrasonic, or optical technologies can be used to track the camera pose as the user moves in the real scene. Barfield and Caudell [9] have provided a comprehensive discussion of the operating principles of the various technologies. Sensor-based tracking systems suffer from some major disadvantages that limit their usefulness for AR applications:

1. These systems are typically very expensive.
2. Extensive infrastructure is required to support the tracking, thus restricting the work area in which an AR system can be deployed.
3. The environment has to be carefully controlled as the sensors are easily affected by perturbations or noise e.g. magnetic sensors are susceptible to electromagnetic interference.

Specifically, AR demands more from trackers and sensors in three areas: greater input variety and bandwidth, higher accuracy, and longer range.

VE systems are primarily built to handle output bandwidth: the images displayed, sounds generated, etc. The input bandwidth is tiny: the locations of the user's head and hands, the outputs from the buttons and other control devices, etc. AR systems, however, will need a greater variety of input sensors and much more input bandwidth. There are a greater variety of possible input sensors than output displays. Outputs are limited to the five human senses. Inputs can come from anything a sensor can detect. Some previous work about this can be found in [10].

The accuracy requirements for the trackers and sensors are driven by the accuracies needed for visual registration. For many approaches, the registration is only as accurate as the tracker. Therefore, the AR system needs trackers that are accurate to around a millimeter and a tiny fraction of a degree, across the entire working range of the tracker. Few trackers can meet this specification, and every technology has weaknesses.

Few trackers are built for accuracy at long ranges, since most VE applications do not require long ranges. Motion capture applications track an actor's body parts to control a computer-animated character or for the analysis of an actor's movements. This is fine for position recovery, but not for orientation. Orientation recovery is based upon the computed positions. Even tiny errors in those positions can cause orientation errors of a few degrees, which is too large for AR systems. Two scalable tracking systems for HMDs have been described in the literature [11] [12]. A scalable system is one that can be expanded to cover any desired range, simply by adding more modular components to the system. This is done by building a cellular tracking system, where only nearby sources and sensors are used to track a user. As the user walks around, the set of sources and sensors changes, thus achieving large working volumes while avoiding long distances between the current working set of sources and sensors. While scalable trackers can be effective, they are complex and by their very nature have many components, making them relatively expensive to construct.

2.1.2 Vision-based Tracking Methods

In recent years, vision-based methods that extract camera pose information from features in the 2D images of the real scene have become increasingly popular for two main reasons:

1. It is convenient and cheap since the 2D images are readily available for a video-based AR system and additional sensors are not required.

2. The camera pose estimates are generally more accurate than those obtained from sensors as the measurement errors are relative to the visually perceived image space units (pixels), not the world space units (meters, inches etc.). Many systems have demonstrated nearly perfect registration, accurate to within a pixel [13] [14] [15].

The basic principles behind these methods are based on the results and theories developed in computer vision and photogrammetry research. Most previous work on the AR registration problem using vision-based techniques can be broadly divided into two categories: the Fiducial Marker Tracking Approach, and the Natural Feature Tracking Approach. More details about the theories and challenges of the vision-based tracking methods will be given in the later parts of this chapter.

2.2 Previous Work

There are at least four classes of potential AR applications have been explored so far: medical visualization, maintenance and repair, annotation, and entertainment. The next section describes work that has been done in each area.

2.2.1 Medical Visualization

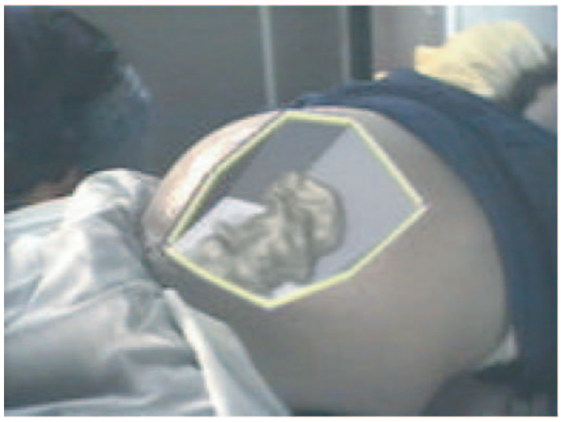

Figure 2.1: Virtual fetus inside womb of pregnant patient. (Courtesy UNC Chapel Hill Dept. of Computer Science.)

Doctors could use Augmented Reality as a visualization and training aid for surgery. It may be possible to collect 3-D datasets of a patient in real time, using non-invasive sensors like Magnetic Resonance Imaging (MRI), Computed Tomography scans (CT), or ultrasound imaging. These datasets could then be rendered and combined in real time with a view of the real patient. In effect, this would give a doctor "X-ray vision" inside a patient. This would be very useful during minimally-invasive surgery, which reduces the trauma of an operation by using small incisions or no incisions at all.

AR might also be useful for training purposes [16]. Virtual instructions could remind a novice surgeon of the required steps, without the need to look away from a patient to consult a manual. Virtual objects could also identify organs

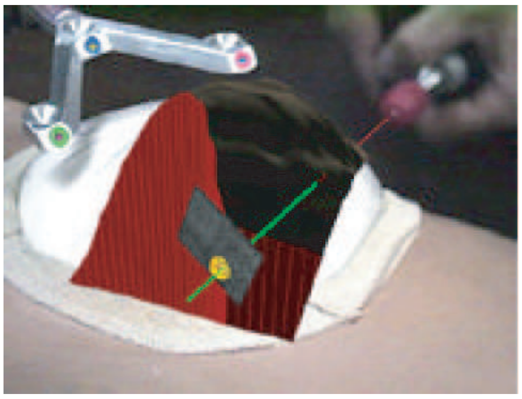

Figure 2.2: Mockup of breast tumor biopsy. 3-D graphics guide needle insertion. (Courtesy UNC Chapel Hill Dept. of Computer Science.)

and specify locations to avoid disturbing [17]. Several projects are exploring this application area. At UNC Chapel Hill, a research group has conducted trial runs of scanning the womb of a pregnant woman with an ultrasound sensor, generating a 3-D representation of the fetus inside the womb and displaying that in a see-through HMD (Figure 2.1). The goal is to endow the doctor with the ability to see the moving, kicking fetus lying inside the womb, with the hope that this one day may become a "3-D stethoscope" [18] [19]. More recent efforts have focused on a needle biopsy of a breast tumor. Figure 2.2 shows a mockup of a breast biopsy operation, where the virtual objects identify the location of the tumor and guide the needle to its target [20]. Other groups at the MIT AI Lab [21] [22] [23], General Electric [24] are investigating displaying MRI or CT data, directly registered onto the patient.

2.2.2 Manufacturing and Repair

Another category of Augmented Reality applications is the assembly, maintenance, and repair of complex machinery. Instructions might be easier to understand if they were available, not as manuals with text and pictures, but rather as 3-D drawings superimposed upon the actual equipment, showing step-by-step the tasks that need to be done and how to do them. These superimposed 3-D drawings can be animated, making the directions even more explicit. Several research projects have demonstrated prototypes in this area. Steve Feiner's group at Columbia built a laser printer maintenance application [25], shown in Figures 2.3 and 2.4. Figure 2.3 shows an external view, and Figure 2.4 shows the user's view, where the computer-generated wireframe is telling the user to remove the paper tray. A group at Boeing is developing AR technology to guide a technician in building a wiring harness that forms part of an airplane's electrical system. Storing these instructions in electronic form will save space and reduce costs. Currently, technicians use large physical layout boards to construct such harnesses, and Boeing requires several warehouses to store all these boards. Such space might be emptied for other use if this application proves successful [26] [27]. Boeing is using a Technology Reinvestment Program (TRP) grant to investigate putting this technology onto the factory floor. Figure 2.5 shows an external view of Adam Janin using a prototype AR system to build a wire bundle at Boeing.

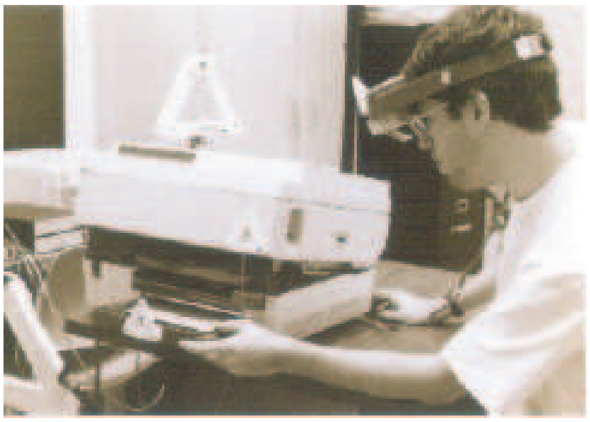

Figure 2.3: External view of Columbia printer maintenance application. Note that all objects must be tracked.

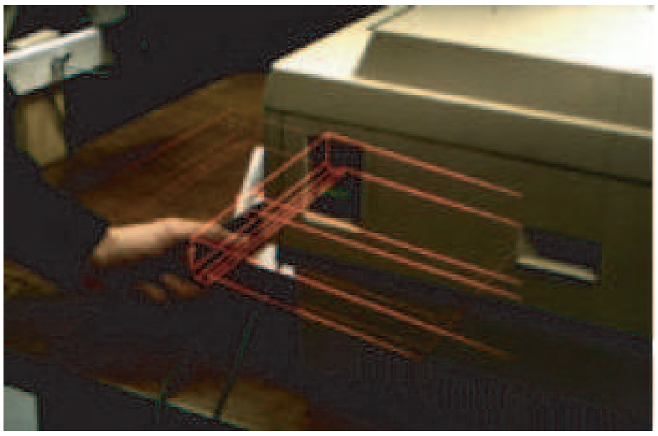

Figure 2.4: Prototype laser printer maintenance application, displaying how to remove the paper tray.

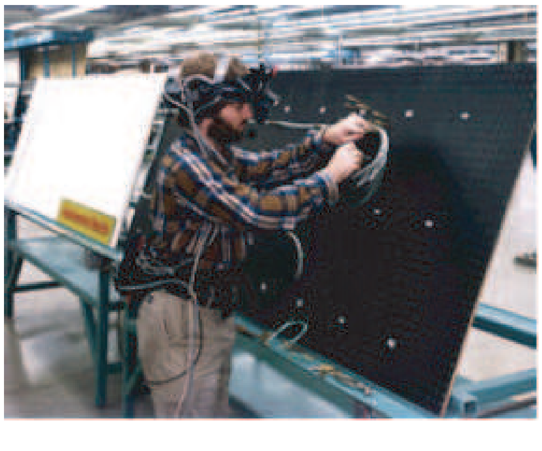

Figure 2.5: Adam Janin demonstrates Boeing's prototype wire bundle assembly application.

2.2.3 Annotation and Visualization

AR could be used to annotate objects and environments with public or private information. Applications using public information assume the availability of public databases to draw upon. For example, a hand-held display could provide information about the contents of library shelves as the user walks around the library [28] [29] [30]. At the European Computer-Industry Research Centre (ECRC), a user can point at parts of an engine model and the AR system displays the name of the part that is being pointed at [31]. Figure 2.6 shows this, where the user points at the exhaust manifold on an engine model and the label "exhaust manifold" appears.

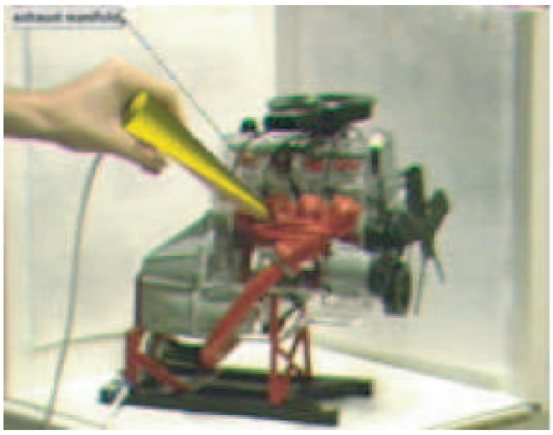

Figure 2.6: Engine model part labels appear as user points at them. (Courtesy ECRC)

2.2.4 Entertainment

In the entertainment sector, several projects have showed "Virtual Sets" that merge real actors with virtual backgrounds, in real time and in 3-D. The actors stand in front of a large blue screen, while a computer-controlled motion camera records the scene. Since the camera's location is tracked, and the actor's motions are scripted, it is possible to digitally composite the actor into a 3-D virtual background. For example, the actor might appear to stand inside a large virtual spinning ring, where the front part of the ring covers the actor while the rear part of the ring is covered by the actor. The entertainment industry sees this as a way to reduce production costs: creating and storing sets virtually is potentially cheaper than constantly building new physical sets from scratch. The ALIVE project from the MIT Media Lab goes one step further by populating the environment with intelligent virtual creatures that respond to user actions [32]. In that system, user's gesture are interpreted by the system based on the context as shown in Figure 2.7.

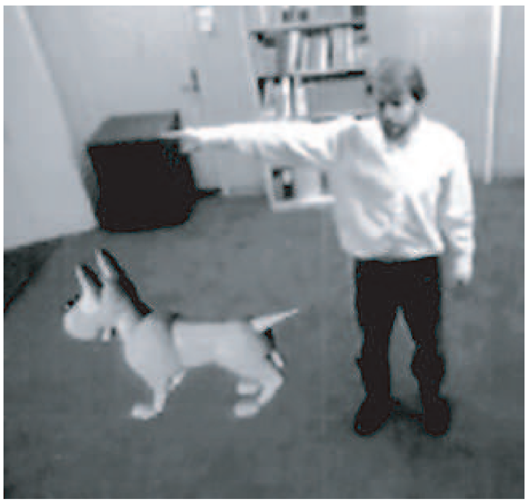

Figure 2.7: The virtual dog is walking away in the direction the user is pointing.

2.3 Challenges in Augmented Reality

2.3.1 AR registration problem

In 1995, Mike Bajura and Ulrich Neumann have pointed out in one of their IEEE paper [13] that registration based solely on the information from the hardware tracking system is like building an "open-loop" controller. The system has no feedback on how closely the real and virtual actually match. Without feedback, it is difficult to build a system that achieves perfect matches. However, video-based approaches can use image processing or computer vision techniques to aid registration. Since video-based AR systems have a digitized image of the real environment, it may be possible to detect features in the environment and use those to enforce registration. They call this a "closed-loop" approach, since the digitized image provides a mechanism for bringing feedback into the system.

This is not a trivial task. This tracking process must run in real time and must be robust. This often requires special hardware and sensors, which varies according to the requirements of the AR application. However, the basic hardware needed for almost all vision-based AR systems are the video capturing devices — cameras, and the displaying devices — usually is the video see-through HMDs.

As just mentioned in Chapter 1, in a typical vision-based AR system as shown in the Figure 1.1, the user views the real book through a video camera on a see-through HMD. Video stream from the camera is combined with the graphic images created by the graphics renderer. The result is then sent to the monitors in front of the user's eyes.

To generate a consistent view of these virtual objects from all views of the real scene so that the illusion that the real and virtual worlds coexist is not compromised, the key requirement is knowledge of the relationships among the object, world and camera coordinate systems (Figure 2.8). This is also commonly known as the AR registration problem. These relationships are determined by the object-to-world, $\mathbf{P}$, world-to-camera, $\mathbf{T}$ and camera-to-image plane, $\mathbf{K}$, transforms [33]. $\mathbf{P}$ specifies the position and orientation of a virtual object with respect to the world coordinate system. The pose or motion of the camera viewing the real scene is defined by $\mathbf{T}$ and is a six-degree of freedom (6DOF) measurement: three degrees of freedom for position and three for orientation relative to the world coordinate

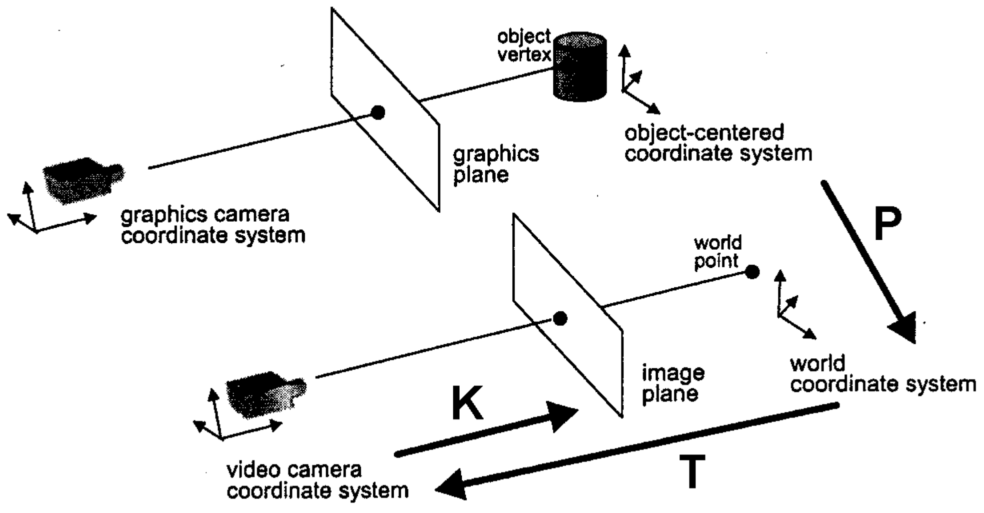
object vertex
object-centered coordinate system
graphics plane
graphics camera coordinate system
P
world point
image plane
world coordinate system
K
video camera coordinate system
T

Figure 2.8: The multiple coordinate systems that must be registered.

system. The projection performed by the camera to create a 2D image of the 3D real scene is specified by **K** which can be obtained by camera calibration. Calculating the camera pose, **T**, for each image frame of the incoming video stream is the main objective of the vision-based tracking algorithm.

2.3.2 Vision-based registration techniques

The basic principles behind the vision-based registration methods are the results and theories developed in computer vision and photogrammetry research. Recently, it becomes an increasingly popular research area. The operational approaches for the vision-based techniques can be broadly divided into two categories: the Fiducial Marker Tracking Approach, and the Natural Feature Tracking Approach.

2.3.3 Fiducial-based Tracking

In this type of registration algorithms, fiducials such as paper markers are placed in the scene where virtual objects are to be introduced (Figure 2.9).

These paper markers have some nice properties such as known shapes and

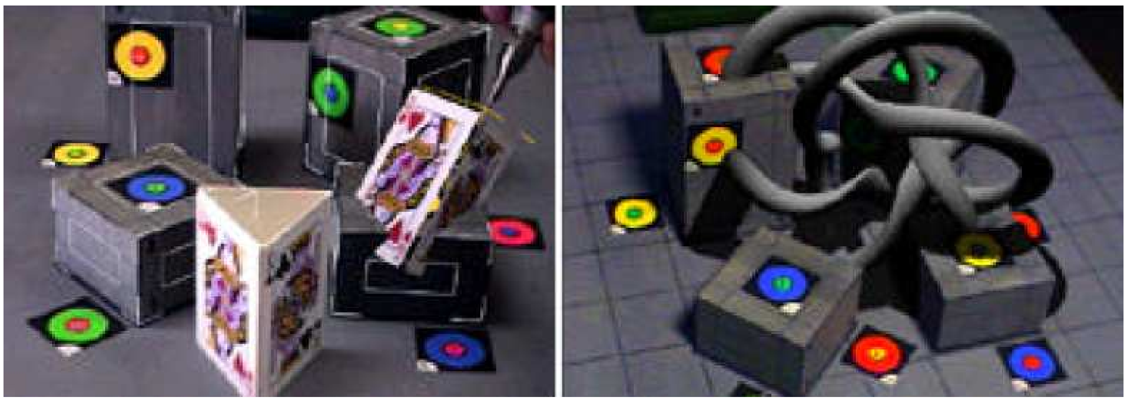

Figure 2.9: Examples of paper markers.

colors which make them easy to detect and identify in the images. In [14], solid-color circle and triangle stickers were used while the AR system in [34] worked with multi-colored concentric ring markers. The centroids of these markers are the features that are tracked in the video stream and at least three markers have to be detected in the image frame before the camera pose can be computed.

The 3D world coordinates of the marker features are measured *a priori* and given the 2D coordinates of the detected features in the images, a correspondence between 3D and 2D is set up. Pose estimation techniques [35][36] can then be used to estimate the camera pose. These markers are inexpensive to produce and the methods are simple and can be implemented in real-time using normal desktop computers. However, camera tracking can be easily lost as it is only based on a few features and there is a limited range of camera viewpoints from which the fiducials are visible.

2.3.4 Natural Feature Tracking

Although the fiducial-based tracking methods can achieve up to sub-pixel accuracy while running realtime, these routines all assume that one or more fiducials are visible at all times; without them, the registration can fall apart.

The more challenging job is to perform camera pose tracking in unprepared environments i.e. no modifications to the environment such as placing fiducial markers or sensors are made. Camera pose measurements are obtained based on naturally occurring features such as corner points and edges in the real scene with *a priori* unknown 3D positions. By using natural features, the tracking range and stability are typically greater than fiducial-based tracking systems since there are more features available to track the camera pose from. Natural feature based tracking systems also allow for AR applications e.g. augmenting video archive footage for special effects, which do not permit the placement of fiducials. Furthermore, the user's visualization of the augmented reality is greatly enhanced since virtual objects are introduced into a completely natural setting. The basic procedure to perform camera pose tracking from natural features involves two main steps:

1. Establishing which features correspond to which between different image frames of the incoming video stream.

2. Estimating the change in camera pose between frames based on the change in 2D positions of these features.

One approach to recover the motion field from the tracking of natural features is reported in [37], where optical flow is used to compute the differential motion estimates between adjacent image frames. However, because it's very computationally heavy to do it this way, it's almost impossible to make it run in realtime on a normal PC.

2.4 Camera Parameters

2.4.1 Pinhole Camera Model

For a vision-based tracking system, the pinhole model is normally used to describe the camera viewing the real scene. Denoting the superscript T as matrix transpose, the perspective projection (Figure 2.8) performed by a pinhole camera that relates the coordinates of a 3D point $\mathbf{P} = [X, Y, Z]^T$ in a user-defined world coordinate system to the corresponding 2D image coordinates $\mathbf{p} = [x, y]^T$ is given by:

$$s\tilde{\mathbf{p}} = \mathbf{K}\mathbf{T}\tilde{\mathbf{P}} \tag{2.1}$$

where $\mathbf{T}$ is a 3×4 Euclidean transformation matrix from the world coordinate system to the camera coordinate system of an image frame. $\mathbf{T}$ is also known as the camera pose. $\mathbf{K}$ is called the camera intrinsic parameter. It is a 3×3 matrix that maps a 3D point expressed in the camera coordinate system to the corresponding 2D image/pixel coordinates. $\mathbf{K}$ is obtained through camera calibration. s is an arbitrary scaling factor and $\tilde{}$ denotes homogeneous coordinates. An inhomogeneous vector $\mathbf{m} = [m_1, m_2, \ldots]^T$ can be transformed into a homogeneous representation by adding a 1 as its last element i.e. $\tilde{\mathbf{m}} = [m_1, m_2, \ldots, 1]^T$. Conversely, given a homogeneous vector $\tilde{\mathbf{m}}$, the inhomogeneous vector is obtained by dividing each component of $\tilde{\mathbf{m}}$ by its last element. The matrix $\mathbf{T}$ can be decomposed as

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \tag{2.2}$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector. Estimating the rotation and translation parameters is the work of the proposed camera pose tracking system. To facilitate the concatenation of Euclidean transformations, the following 4×4 matrix $\breve{\mathbf{T}}$ is defined:

$$\breve{\mathbf{T}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

2.4.2 Camera Calibration

Camera calibration refers to the estimation of the intrinsic properties of the camera. For a pinhole camera model, the intrinsic parameters are described by a 3×3 matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} s_x f & k & x_o \\ 0 & s_y f & y_o \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

where f is the focal length, s_x is the scale factor (pixel/mm) in direction of x axis, s_y is the scale factor in direction of y axis and (x_o, y_o) is the pixel coordinates of the image center. k is a skew factor or slant between the x-axis and y-axis and is usually very small.

The calibration algorithm in the ARToolKit software [38], an open source library for developing computer-vision-based AR applications, is used to estimate the intrinsic parameter matrix $\mathbf{K}$. An image of a simple cardboard with a ruled grid of lines (Figure 2.10) is captured. 3D coordinates of all cross points of the line grid are known in the cardboard local coordinate system and the corresponding 2D image coordinates can be detected by image processing techniques. Similar to Equation 2.1, the perspective relationship between the image coordinates (x_c, y_c)

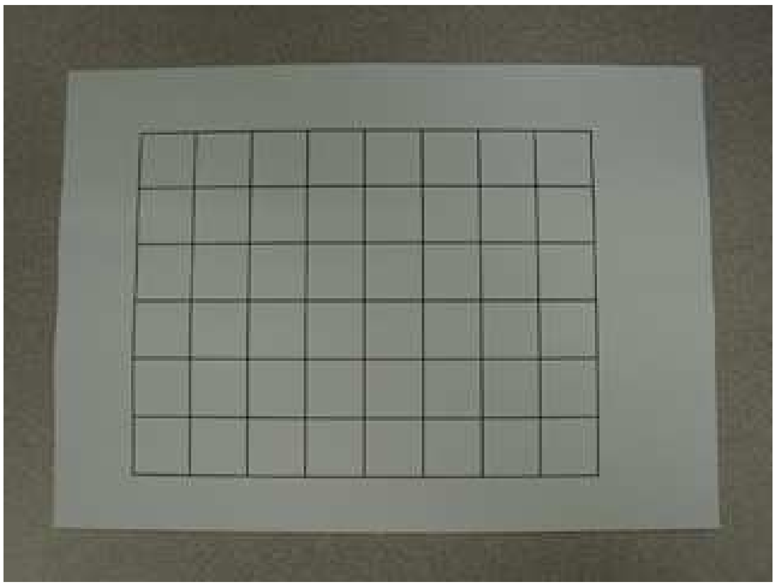

Figure 2.10: Image of a known calibration pattern.

and the card coordinates (X_w, Y_w, Z_w) is represented as:

$$s \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{T_c} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.4}$$

where $\mathbf{T}_c$ represents the translation and rotation transformation from the cardboard coordinates to the camera coordinates and has the same form as Equation 2.2. s is an arbitrary scaling factor. From Equation 2.4, each pair of matched (x_c, y_c) and (X_w, Y_w, Z_w) gives rise to two linearly independent equations in the unknown parameters of $\mathbf{K}$ and $\mathbf{T}_c$. Since many pairs of (x_c, y_c) and (X_w, Y_w, Z_w) have been obtained, the unknown parameters can be solved for.

Chapter 3

Tangible AR Interface Development

Although the Augmented Reality (AR) technology has come a long way from rendering simple wireframes in the 1960s [5], AR interface design and interaction space development have only had limited progress so far. The previous work in this area includes Feiner's MARS Authoring Tool [39], Piekarski's Tinmith-Metro mobile outdoor modelling application [40], Mark Billinghurst and Kato's Magic Book [8]. Although researchers and developers have made great advances in display and tracking technologies, but interaction with AR environments has been largely limited to passive viewing or simple browsing of virtual information registered to the real world.

To overcome these limitations, in this thesis work, we seek to design an AR interface that provides users with interactivity so rich it would merge the physical space in which we live and work with the virtual space in which we store and interact with digital information. In this single augmented space, computer-generated entities would become first-class citizens of the physical environment. We would use these entities just as we use physical objects, selecting and manipulating them with our hands instead of with a special-purpose device such as a mouse or joystick. Interaction would then be intuitive and seamless because we would use the same tools to work with digital and real objects.

3.1 Tangible AR Interface

Tangible interfaces are powerful because the physical objects used in them have properties and physical constraints that restrict how they can be manipulated and so are easy to use. However there are limitations as well. It can be difficult to change these physical properties, making it impossible to tell from looking at a physical object what is the state of the digital data associated with that object. In some interfaces there is also often a disconnect between the task space and display space. For example, in the Gorbet's Triangles work, physical triangles are assembled to tell stories, but the visual representations of the stories are shown on a separate monitor distinct from the physical interface [41].

The visual cues conveyed by tangible interfaces are also sparse and may be inadequate for some applications. The ToonTown remote conferencing interface uses real dolls as physical surrogates of remote people [42]. However the non-verbal and visual cues that these objects can convey is limited compared to what is possible in a traditional video conference. Showing three-dimensional imagery in a tangible setting can also be problematic because it is dependent on a physical display surface.

Many of these limitations can be overcome through the use of Augmented Reality. We define Tangible Augmented Reality as AR interfaces based upon Tangible User Interface design principles. In these interfaces the intuitiveness of the physical input devices can be combined with the enhanced display possibilities provided by virtual image overlays. Head mounted display (HMD) based AR provides the ability to support independent public and private views of the information space, and has no dependence on physical display surfaces. Similarly, AR techniques can be used to seamlessly merge the display and task space.

Research in immersive virtual reality point to the performance benefits that can result from a Tangible Augmented Reality approach. The physical properties of the tangible interface can be used to suggest ways in which the attached virtual objects might interact and enhance the virtual interaction. For example, Lindeman finds that physical constraints provided by a real object can significantly improve performance in an immersive virtual manipulation task [43]. Similarly Hoffman finds adding real objects that can be touched to immersive Virtual Environments enhances the feeling of presence in those environments [44]. While in Poupyrev's virtual tablet work, the presence of a real tablet and pen enable users to easily enter virtual handwritten commands and annotations [45].

3.2 The Design Approach

Interfaces that combine Reality and Virtuality are not new. However, Ishii summarizes the state of AR research when he says that AR researchers are primarily concerned with "the purely visual augmentation" rather than the form of the physical objects those visual augmentations are attached to [46]. If we are to create more usable AR interfaces then researchers must have a better understanding of design principles based on form as well as function.

In our augmented reality work we advocate designing the form of physical objects in the interface using established Tangible User Interface design methods. Some of the tangible design principles include:

- Object affordances should match the physical constraints of the object to the requirements of the task.

- The ability to support parallel activity where multiple objects or interface elements are involved at once.
- Support for physically based interaction techniques (such as using object proximity or spatial relations).
- The form of objects should encourage and support spatial manipulation.

Physical interface attributes are particularly important in interfaces designed to support face-to-face collaboration. In this case people commonly use the resources of the physical world to establish a socially shared meaning. Physical objects support collaboration both by their appearance, the physical affordances they have, their use as semantic representations, their spatial relationships, and their ability to help focus attention. In an AR interface the physical objects can further be enhanced in ways not normally possible such as providing dynamic information overlay, private and public data display, context sensitive visual appearance, and physically based interactions. In the next section we describe how the Tangible Augmented Reality approach was applied in an early collaborative table-top AR experience.

3.3 Implementing Global Coordinate Tracking

An augmented-reality system's fundamental elements include techniques for tracking user position and viewpoint direction, registering virtual objects relative to the physical environment, then rendering and presenting them to the user. In this thesis work, we implemented the system based on the open source ARToolKit software [47], which provides methods for tracking fiducial markers.

To create the physical fiducial markers, we use paper cards measuring $15cm \times 15cm$ with simple square patterns consisting of a thick black border and unique symbols in the middle. We can print any symbol for identification as long as each symbol is asymmetrical enough to distinguish between the square border's four possible orientations. An example of such fiducial marker is shown in Figure 3.1.

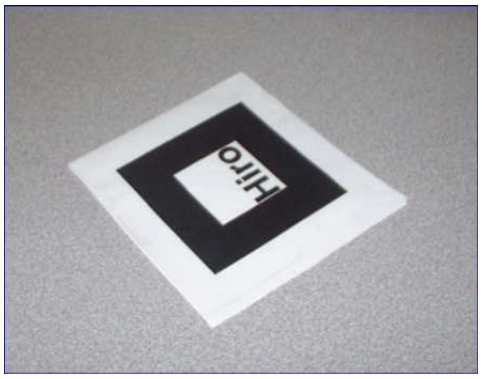

Figure 3.1: An example of the fiducial marker used in the tangible AR interaction application.

The system captures the camera's video stream at 640×480 pixel resolution. By tracking rectangular markers of known size, the system can find the relative camera position and orientation in real time and can then correctly render virtual objects on the physical cards.

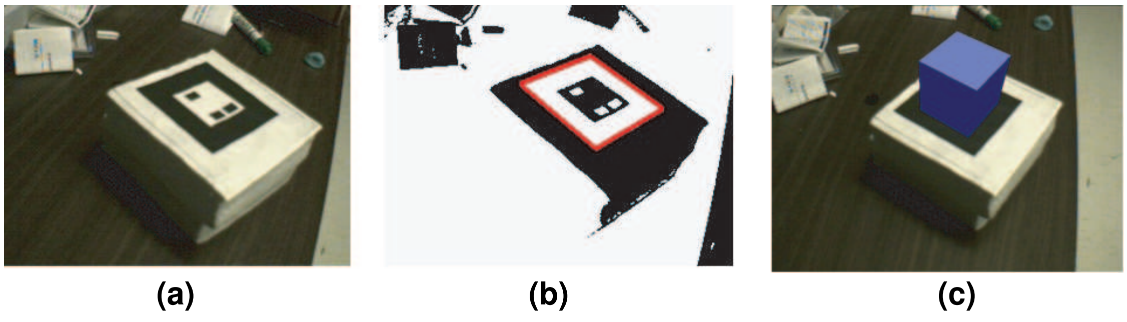

Figure 3.2: The three-step process of mapping virtual objects onto physical fiducial markers so that the user can view them with a head-mounted display (HMD).

ARToolKit uses computer vision techniques to calculate the real camera viewpoint relative to a real world marker. There are several steps as shown in Figure 3.2. First the live video image (Figure 3.2-a) is turned into a binary (black or white) image based on a lighting threshold value (Figure 3.2-b). This image is then searched for square regions. ARToolKit finds all the squares in the binary image, many of which are not the tracking markers. For each square, the pattern inside the square is captured and matched against some pre-trained pattern templates. If there is a match, then ARToolKit has found one of the AR tracking markers. ARToolKit then uses the known square size and pattern orientation to calculate the position of the real video camera relative to the physical marker. A 3×4 matrix is filled in with the video camera real world coordinates relative to the card. This matrix is then used to set the position of the virtual camera coordinates. Since the virtual and real camera coordinates are the same, the computer graphics that are drawn precisely overlay the real marker (Figure 3.2-c). The OpenGL API is used for setting the virtual camera coordinates and drawing the virtual images. The details on this fiducial marker tracking algorithm can be fount in [47].

3.4 Tracking of Multiple Fiducials

The tracking method in ARToolKit provides satisfactory accuracy for a table-top AR environment, however it uses a single relatively large square marker as a fiducial. So if a hand or other object to even partially overlapped the fiducial the tracking was lost. This decreased the robustness of tracking under the conditions where a hand could overlap the fiducials. Also if there is some distance between tracked fiducials and displayed virtual objects, tracking errors strongly influence the registration accuracy. That is, using a single fiducial decreases the accuracy of registration under the conditions where virtual objects need to be displayed around on the table.

To solve this problem, Mark Billinghurst and Kato have tested a new algorithm to track multiple fiducials in their Magic Book project [8]. In this thesis work, I have also developed a tracking method in which multiple markers are used as fiducials and pose and position are estimated from all of the detected fiducial marks. This means that many of the fiducial, as long as not all of them, can be covered up without losing tracking. This is critically important, as in a typical tangible AR application, it's not avoidable that the user's hand will cover some of the fiducials when he/she is interacting with the virtual objects.

When there is a fiducial marker in the image, it is possible to estimate 3D pose and position using our earlier method in ARToolKit. However if there is more than one fiducial markers are visible, we can achieve more robust tracking if we estimate pose from all of available features. In order to do this we adopt following procedures:

(step 1) The biggest fiducial marker is selected in the image. 3D pose and position are initially estimated from it. This information is represented as the following transformation function from marker coordinates to camera coordinates:

$$(x_c, y_c, z_c) = trans(x_w, y_w, z_w) \tag{3.1}$$

where $(\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w)$ is a position in world coordinates and $(\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$ is the same position in camera coordinates.

(step 2) The positions of all other fiducial markers are then estimated in screen coordinates by using the above transformation function, a projective function and the 3D positions of these markers in the world coordinates:

$$(x_s, y_s) = perspect(trans(x_w, y_w, z_w)) \tag{3.2}$$

where the function *perspect* is a projective function. This function consists of perspective projection parameters and image distortion parameters.

(step 3) The actual screen coordinates of the detected markers are compared to the estimated positions. Using the positions of all successfully matched fiducial markers, the 3D pose and position are re-estimated. For this calculation, the initial transformation function is used and modified as the amount of errors between the actual feature positions in the image and the estimated positions goes to minimum using a hill-climbing method.

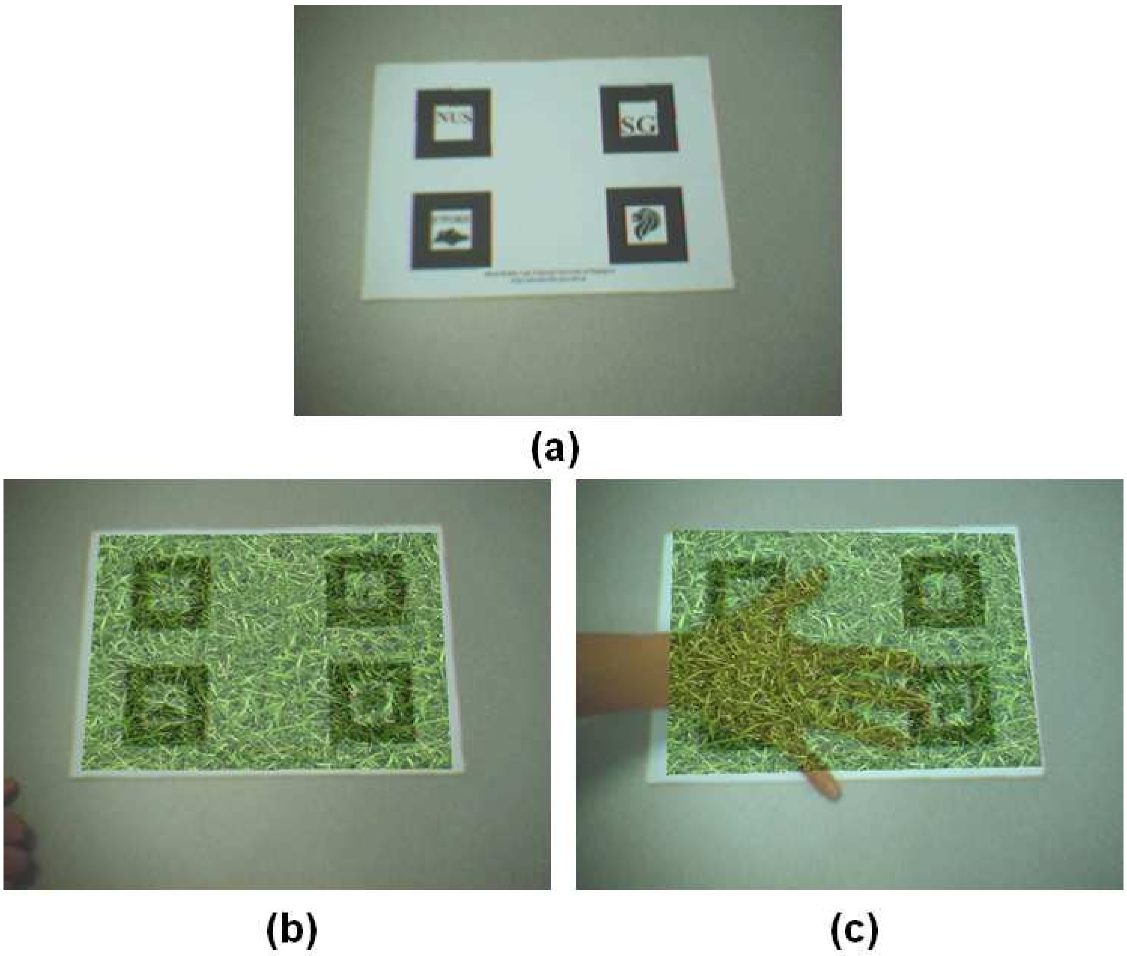

Figure 3.3: Tracking of multiple fiducials. (a) The four different fiducials are printed on the same paper. (b) A virtual grass is augmented onto the fiducials. (c) Even some of the fiducials are blocked, the correct registration still remains.

The result of this proposed method is shown in Figure 3.3. As the user moves his hand, and covers some of the fiducials on the desk, the virtual object augmented on the table surface is still correctly registered. However, because only one camera is used, the depth information is not available, problem with incorrect occlusion results. As shown in Figure 3.3-c, the virtual grass incorrectly appears in front of the hand. It is possible to solve this problem by getting depth information from stereo cameras. However, it will also increase the computational load. Because of the objective of this thesis work is to build a realtime AR system for normal desktop PC, or even laptop computers and small wearable computers, we decided to take the mono-view approach.

3.5 Implementing Natural and Intuitive Manipulation

Based on the tracking methods described above, applications for Augmented Reality can be developed to create Tangible User Interfaces. Real world objects (like cards with fiducial markers printed on top) can be manipulated by the users, with virtual objects superimposed upon them. Manipulation of these Real world objects can then be used to interact with the computer in a natural way.

In Mark Billinghurst and Kato's Magic Book project [8], users were encouraged to interact with the virtual objects by a natural way without touching the mouse and keyboard. In this thesis work, I also adopted a similar approach. The relative distance between the centers of the two markers, as well as the angle between the two marker surfaces, are calculated in real time. Corresponding actions are then applied to such situation. A simple application of such a tangible interaction in this thesis work is the *Picking* and *Dropping* of the virtual object by controlling

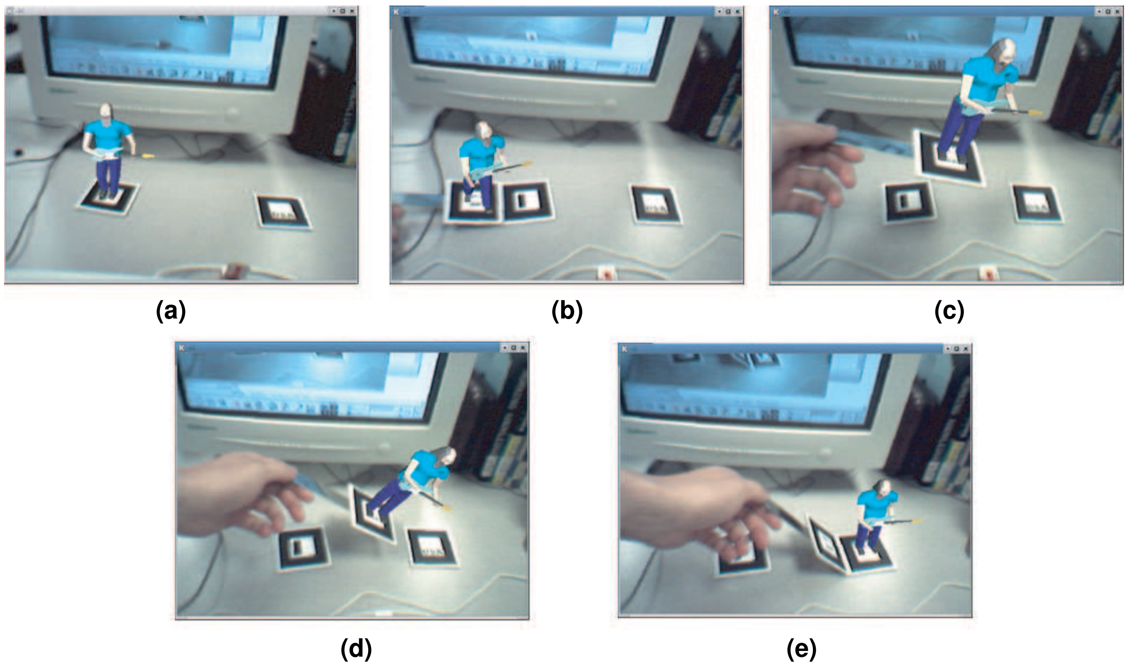

Figure 3.4: Tangible Interaction: Picking and Dropping virtual objects by manipulating physical cards.

the distance and the angle between the markers (Figure 3.4).

- *Picking* is defined when the fiducial marker **A** (the transportation paddle) is been moved very closely to the fiducial marker **B** (the original position for the 3D object), while remaining horizonal comparing to marker **B**, that is, angle between these two marker surfaces is small enough. When *Picking* happens, the system will stop rendering the 3D object on the fiducial marker **B**, but on the marker **A** instead (Figure 3.4-b).

- After *picking* up the virtual object, the user can then move it around the environment, or even move it near to his eyes, which is the camera on his HMD now, to have a close look of these virtual characters.

- *Releasing* or *Dropping* is defined when the fiducial marker **A** (the transportation paddle) is been moved very closely to the fiducial marker **C** (the destination for the 3D object), and tilt it up to a certain angle (e.g. 50 degrees) relative to the surface defined by marker **C**. When it happens, the system will stop rendering the 3D object on the fiducial marker **A**, but on the marker **C** instead (Figure 3.4-e).

3.6 Applications for Tangible Interfaces

A more sophisticated application of these ideas is presented in a prototype desktop AR system *Virtual Hello Kitty Garden* (Figure 3.5). Here, a catalogue of three dimensional cartoon models is created by attaching a different fiducial marker to each page of a real book (Figure 3.5-a). Similar to Mark and Kato's Magic Book project, users of this Virtual Hello Kitty Garden system can select the model that they are interested in by simply turning the pages (Figure 3.5-b). Again, this is a radical departure from scrolling through a list of three dimensional models on a computer screen. The interface is the real-world object which is a book in this case. The user also has a virtual paddle (Figure 3.5-c). When he brings the paddle adjacent to the three dimensional object, the object is copied to the paddle (Figure 3.5-d). It can then be moved through the environment. The object can be released and placed on a surface by tilting the paddle relative to that surface (Figure 3.5-e). In this way, the user can place the virtual objects on a virtual surface to create a model garden (Figure 3.5-f).

The multiple fiducial tracking approach is used for the registration of the virtual garden. Four different markers are used in this case (Figure 3.5-e). So, even if the hand of the user blocks some of the markers when he picks up or releases the virtual Hello Kitty, the position of the garden is still accurately tracked. This tracking technique is experimented to be robust enough to correctly track patterns without losing performance. We implemented this virtual Hello Kitty Garden system on

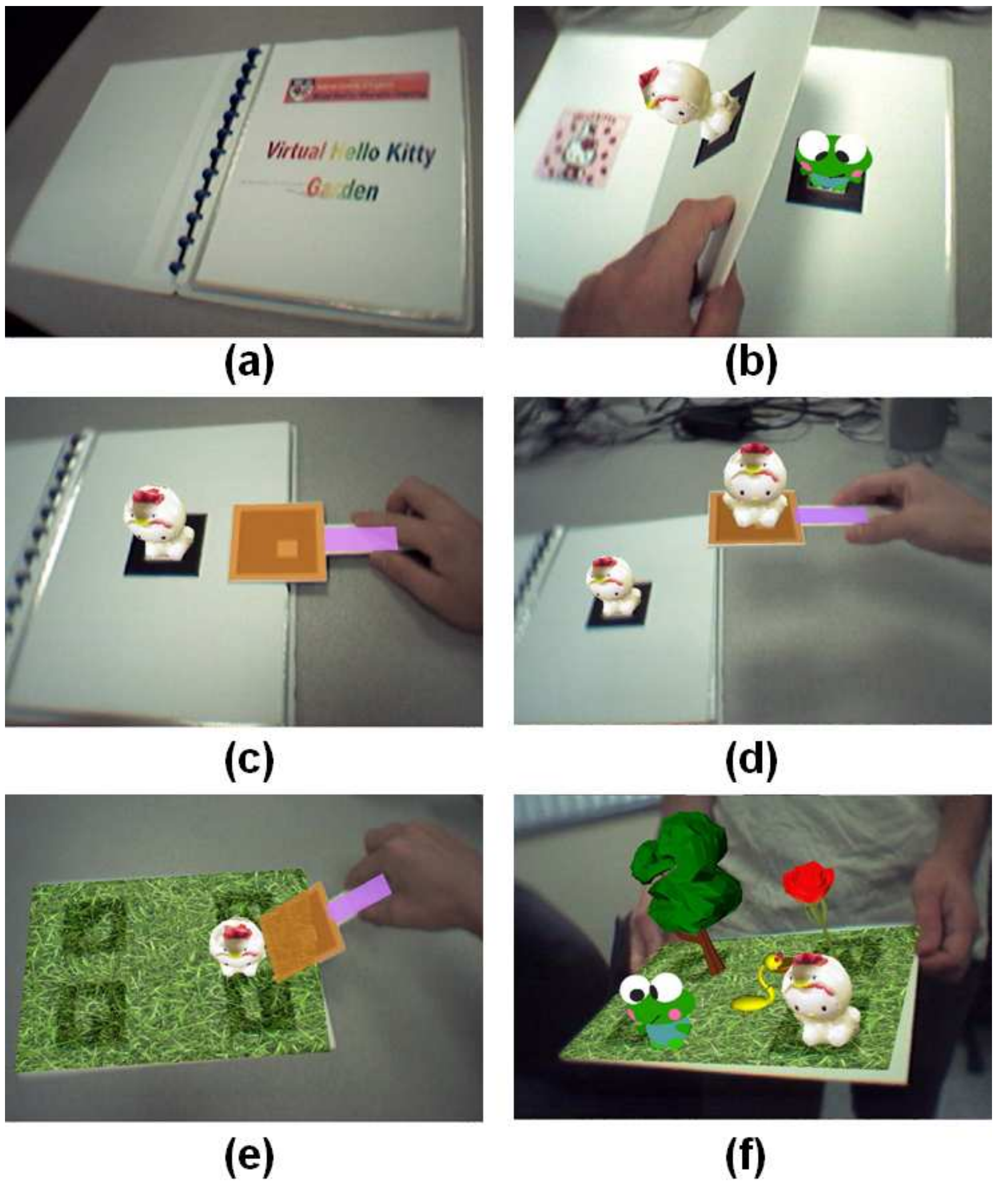

Figure 3.5: A More Sophisticated Tangible Interaction Example: Virtual Hello Kitty Garden.

a normal Pentium III PC running Linux, which allows updates at 30 frames per second. A short video clip TangibleInteraction.avi can be found in the attached CD-ROM.

Chapter 4

Tracking in Unprepared AR Environments

In fiducial-based AR systems, the positions and/or shapes of these markers are known in advance. So, it is relatively easy to establish the position of the camera relative to the scene and introduce the virtual content.

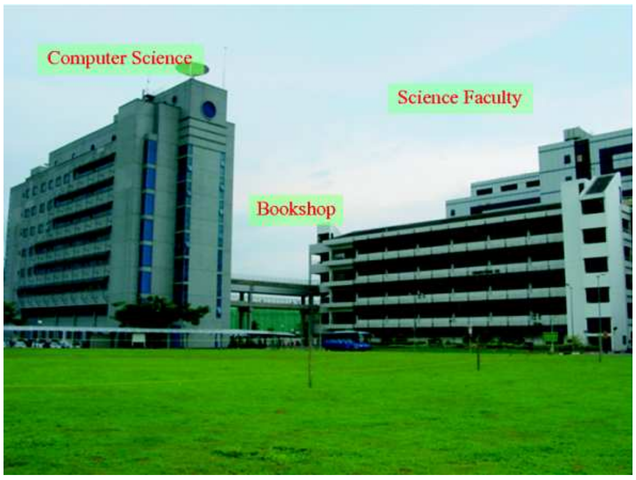

Figure 4.1: Geographic labeling refers to the real-time annotation of outdoor scenes via augmented reality displays.

In the second part of this thesis work, I attempt to replace these fiducial markers with "natural feature tracking". This is particularly important for the applications where the environment is necessarily unprepared, e.g. the outdoor geographical labeling applications [48] [49] [50] (Figure 4.1). The aim of these applications is to calculate the image motion field between two different pictures of the same geographical object. If we know where the text label should go in the first image, then the motion field tells us where to place it in the second image. In this thesis work, two algorithms are introduced which compute highly reliable approximations to this motion field.

Most current techniques are based on an initial optical flow calculation. For example, Neumann's work in tracking in natural environments [37] [51]. However, there are a number of disadvantages of such techniques. Optical flow calculation is not robust and may give erroneous velocity estimates. It can also only measure small velocities and is not well-suited for large image motion such as that created by camera rotation. Finally, and most importantly, optical flow calculation does not take into account global geometric constraints in the image flow field (see Figure 4.2). Geographical annotation applications involve a camera moving in a (mostly) static, rigid environment. Under these circumstances image flow is constrained by epipolar geometry. When the camera motion is a pure rotation or the scene is planar, then even more strict constraints are placed on the motion flow field.

In the following sections we describe these mathematical constraints on image motion. Then we discuss how to estimate these flow constraints before considering how this information can be applied to various indoor and outdoor AR applications, e.g. geographic labeling. We then provide a detailed analysis of the advantages and limitations of these techniques and how to combine them into a full wearable computing application in Chapter 6.

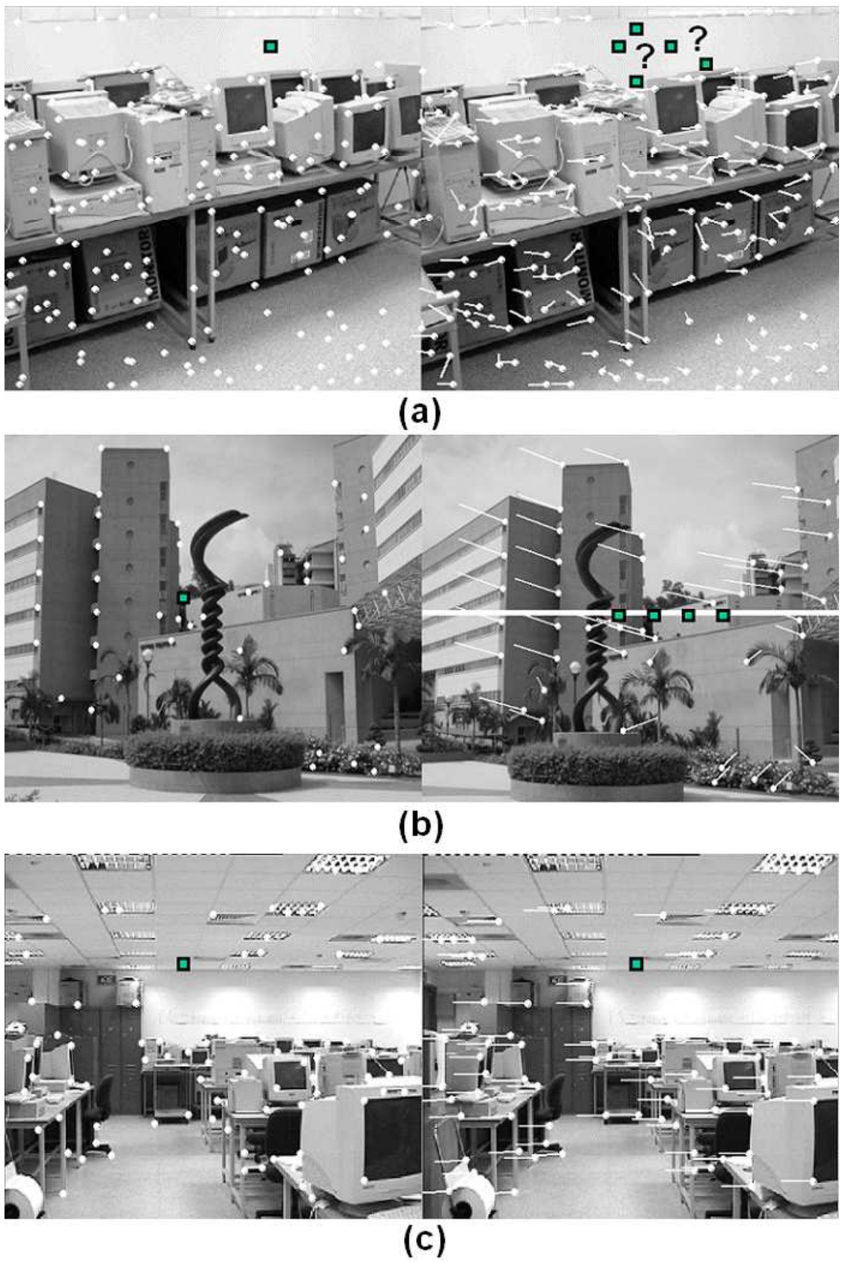

Figure 4.2: Image motion constraints. In each case we are attempting to calculate the motion or flow between the left and right images. (a) Results of optical flow calculation. A noisy estimate of the image movement is detected independently at each corner point. A given point such as the one denoted by a square, can map to anywhere in the second image. (b) Epipolar Constraint. For arbitrary movement in a static scene, a given point in the first image is constrained to lie on a line in the second image. The mapping from point to line is described by the fundamental matrix. (c) In certain cases, the image flow is well described by a homography. This maps a point in the first image to an unique point in the second image.

4.1 Image Motion Constraints

It is common to restrict the estimation of the velocity field to particular "features of interest" in the image. These are chosen to exhibit stability when viewed from different angles, and to provide unambiguous motion estimates. The general approach is to estimate the motion of these points and then interpolate between these known motions to estimate the velocity at general points in the image. This technique also has the benefit that it reduces the image motion estimation problem to establishing matches for just a few points, which is computationally less intensive than trying to estimate a dense velocity map.

4.1.1 Detection of the Feature Corners

A set of feature points or corners corresponding to high curvature points is extracted from both the reference image and the current image. There are a number of algorithms available for selecting these points of interest (see [52] for a review). For this application, OpenCV's implementation of the Harris corner detector [53, 54] is used. Consider the following autocorrelation matrix of image derivatives:

$$\mathbf{C} = \begin{bmatrix} \sum D_x^2 & \sum D_x D_y \\ \sum D_x D_y & \sum D_y^2 \end{bmatrix} \tag{4.1}$$

where D denotes the gray level pixel intensity and D_x and D_y indicate the x and y directional derivatives respectively. The sums are taken over a small neighborhood about an image point. Geometrically, the eigenvectors of the matrix $\mathbf{C}$ encode edge directions and the eigenvalues, λ_1 and λ_2, edge strength. A corner is identified by two strong edges; therefore, if $\lambda_1 \geq \lambda_2$, a corner is a location where the smaller eigenvalue, λ_2, is sufficiently large.

The function `cvGoodFeaturesToTrack` in OpenCV [53] first calculates the minimal eigenvalue $\lambda_2(x, y)$ for every image pixel based on equation 4.1 and retains the local maxima in each 3×3 neighborhood. Denoting the largest $\lambda_2(x, y)$ in the set of retained corners as $\lambda_2(max)$, corners with $\lambda_2(x, y) < q\lambda_2(max)$ are then rejected. Here q stands for the quality level multiplier for the maximum eigenvalue. It specifies minimal accepted quality of image corners. Finally, the function ensures that all the corners are distanced enough from one another by getting two strongest features and checking that the distance between the points is satisfactory. The quality parameter q and the distance threshold (it is the minimum possible distance between returned corners) are set to 0.05 and 20 respectively.

This method allows the recovery of a corner position up to pixel precision. For sub-pixel accuracy, the corner positions are refined using the `cvFindCornerSubPix` function in OpenCV. The methodology used is explained in the manual of OpenCV.

4.1.2 The Epipolar Constraint

Consider a camera viewing the same scene from two different angles (see Figure 4.3). Given the image of a point in the first camera, we only know its direction from the optical centre, and not its depth. Hence, it may lie anywhere along a line in space. One must inevitably conclude that the image of the point in the second image must lie somewhere along the projection of this line in the second camera. Every such line in 3-D space must pass through the optical centre of camera one, and hence every projected line in the second camera must pass through the projection of the optical centre of the first camera. This is termed the epipole, and the line $\mathbf{l}'$ is commonly known as the epipolar line. This epipolar geometry is symmetric, meaning that for a corner $\mathbf{p}'$ in the second image, the corresponding corner $\mathbf{p}$ in the first image must also lie on an epipolar line.

These constraints can be expressed mathematically by the fundamental matrix, $\mathbf{F}$. It can be shown that (homogeneous) points in image 1 and image 2 are related by the 3×3 matrix, $\mathbf{F}$ such that:

$$\mathbf{x'^T F x} = \mathbf{0} \tag{4.2}$$

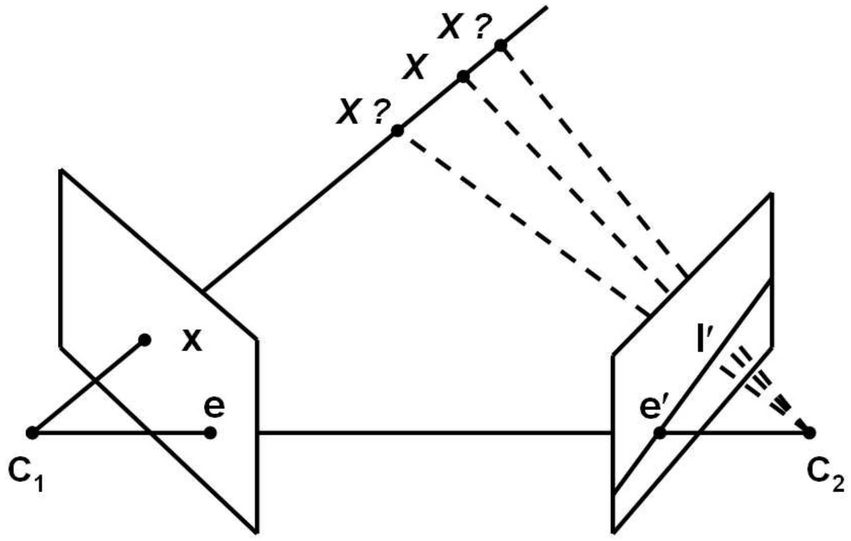

Figure 4.3: Epipolar Constraint. Consider two cameras viewing the same scene from different positions. A point in one image must lie somewhere along the line projecting through the optical centre. The position in the second image is constrained to lie on the projection of this line which is known as an epipolar line. This mapping from points to lines is described by the fundamental matrix.

When the camera calibration and motion information are available, the fundamental matrix $\mathbf{F}$ has the following form:

$$\mathbf{F} = \mathbf{K}^{-T}[\mathbf{t}]_{\times}\mathbf{R}\mathbf{K}^{-1} \tag{4.3}$$

where $\mathbf{K}$ is the intrinsic parameter matrix of the calibrated camera, $(\mathbf{R}, \mathbf{t})$ is the 3D displacement (rotation and translation) from the first to the second camera coordinate system. $[\mathbf{t}]_\times$ is a skew-symmetric matrix of the translation vector $\mathbf{t}$. For a vector $\mathbf{a} = [a_1, a_2, a_3]^T$, the corresponding skew-symmetric matrix is defined as follows:

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{4.4}$$

The fundamental matrix, $\mathbf{F}$, maps any point in one image to a line in the second image. This reduces the search for the correct velocity to a one-dimensional problem. This constraint can be used as a criterion for the rejection of plausible, but false matches.

4.1.3 The Homography Constraint

Under some special circumstances, image motion between two views of the same rigid scene is further constrained by another relationship called *Homography.* A homography is a one-to-one mapping between two images, which is defined by only eight parameters. This model exactly describes the image motion between two frames of a video sequence when

- the camera motion is pure rotation, or
- the camera is viewing a planar scene.

Usually, feature displacement between two images depends on both the camera movement and the camera's distance from the feature. A simple parameterized mapping is therefore not possible. However, in many circumstances, the homography represents a good approximation of the true image flow, particularly when the image structure is near planar, or the camera movement is small and the scene structure is mostly distant.

Consider a set of points in the first image of a sequence with homogeneous coordinates $\mathbf{x} = [x, y, z]^T$, which are known to map to a set of points in the second image, $\mathbf{x}' = [x', y', z']^T$. The relationship between the two images is a homography if the following equation holds:

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \tag{4.5}$$

In other words, the homography, $\mathbf{H}$, maps coordinate $\mathbf{x}$ to coordinate $\mathbf{x}'$. Note that these are homogenous coordinates — each point on the screen is treated as a ray through the camera center. We find the actual image position by dividing the first and second components by the third. The homography is, therefore, a simple linear transformation of the rays passing through the camera center. Roughly speaking, the homography can encompass rotations, scaling, and shearing of the ray bundle.

Figure 4.4 presents another way to consider this concept. Instead of transforming the rays, we can equivalently transform the camera plane onto which the rays are projected. The diagram shows the cube's rays passing to the camera center, $\mathbf{O}$. The three image planes, $\mathbf{P_1}$, $\mathbf{P_2}$, and $\mathbf{P_3}$, intersect these rays in three different ways. The resulting images are related to one another by projective transformations or homographies. A characteristic property of homography is that it always maps a straight line to another straight line, but parallelism is not necessarily preserved.

Although the matrix describing the homography contains nine elements, it's ambiguous up to scale — the use of homogeneous coordinates means that any multiple of the homography will have the same effect. Consequently, because there are only eight independent elements, we can measure the homography relating two images using any four general point correspondences. Each point correspondence generates two linear equations. To solve for the elements of $\mathbf{H}$, we note that $\mathbf{x}'$ and

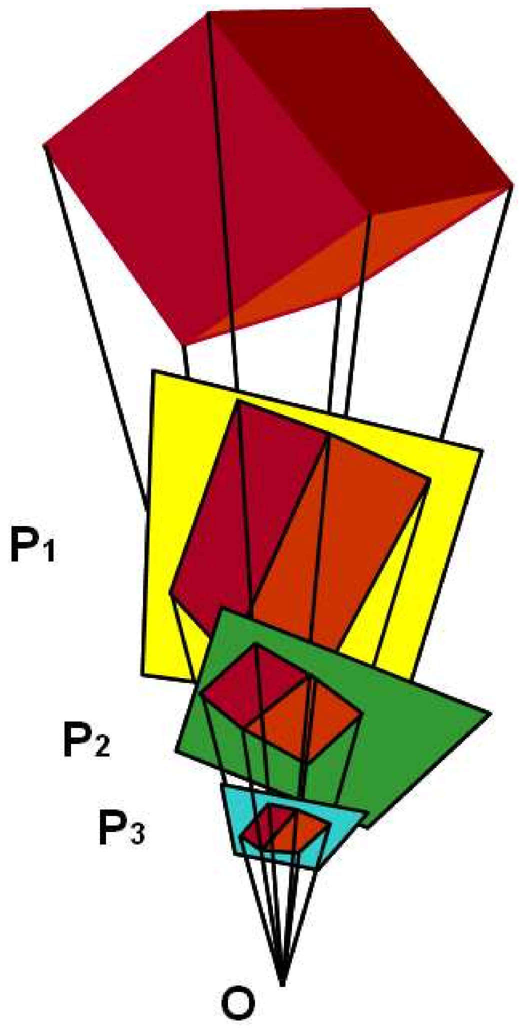

Figure 4.4: Geometric representation of a planar projective transformation of homography. The images on different camera planes cutting the same ray bundle are related by homographies.

Hx are, by definition, rays pointing in the same direction. Their cross product is equal to zero:

$$\mathbf{x}' \times \mathbf{H}\mathbf{x} = \begin{bmatrix} y'\mathbf{h}_3^{\mathrm{T}}\mathbf{x} - z'\mathbf{h}_2^{\mathrm{T}}\mathbf{x} \\ z'\mathbf{h}_1^{\mathrm{T}}\mathbf{x} - x'\mathbf{h}_3^{\mathrm{T}}\mathbf{x} \\ x'\mathbf{h}_2^{\mathrm{T}}\mathbf{x} - y'\mathbf{h}_1^{\mathrm{T}}\mathbf{x} \end{bmatrix} = 0 \tag{4.6}$$

where $\mathbf{h_3}$ is the third row of the homography matrix, **H**. This set of equations provides two independent linear constraints on the components of **H**. If we find four general point correspondences, we can provide eight equations to solve for the eight unknowns of the homography. Generally, we aim to find more point correspondences and calculate an overdetermined least-squares solution. The error in the point positions is linear, but these terms are quadratic in the set of equations. Ideally, we should use this solution as an initial estimate in a subsequent nonlinear minimization of the Euclidean projection error.

Planar scene structure

The homography completely describes the relationship between any two ideal images of a planar structure (such as a notice board). Figure 4.5 shows why this is the case, in which two cameras view the same board from different angles. Consider the rays projecting outward from camera center, $\mathbf{O_1}$. Because the notice board and the image plane 1 both intersect the same ray bundle, they must be related by a homography, which we denote $\mathbf{H_1}$. A similar argument applies to the relationship between the notice board and image plane 2, which the homography $\mathbf{H_2}$ describes. Because homographies form a closed group under multiplication, the two images of the board must be related by a third homography $\mathbf{H_2^{-1}H_1}$.

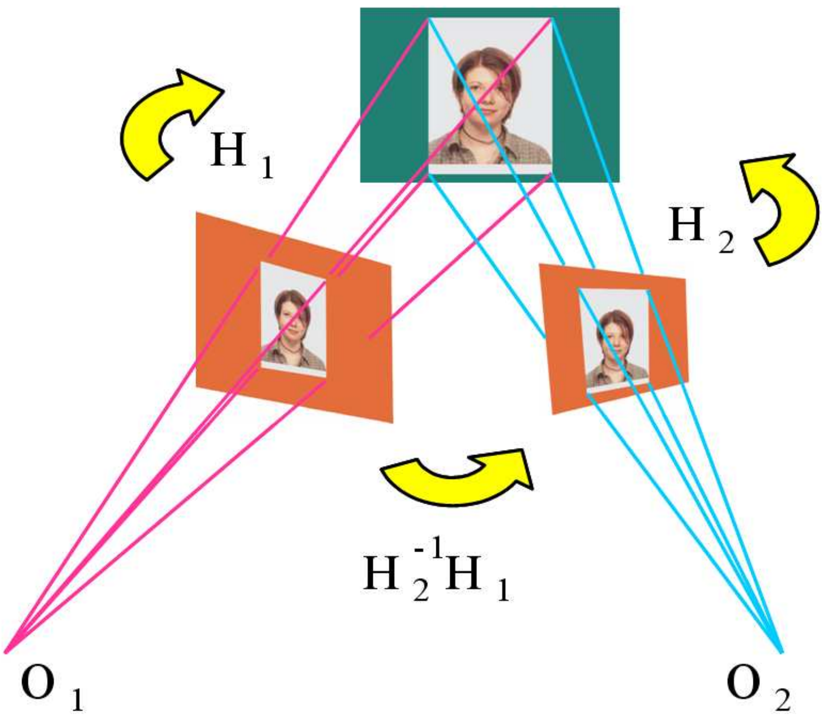

Figure 4.5: Any two views of a planar scene are related by a homography.

Pure camera rotation

Figure 4.6 shows why a homography can describe, in addition to planar structure, pure camera rotation. The left image shows a camera viewing a 3D cube. When the camera rotates, the image plane simply cuts the rays at a different angle. Rotations thus form a subset of the transformations described in Figure 4.4. This knowledge allows us to apply natural feature tracking, based on homographies, to the problem of registering information in outdoor scenes. Such a problem has previously been attacked through various methods, including inertial trackers, 2D computer vision, and hybrid approaches. We argue that outdoor tracking requirements are frequently satisfied by calculating a homography to a stored reference frame. Because most objects in this setting are distant, we can consider camera movement to be approximately a fixed rotation.

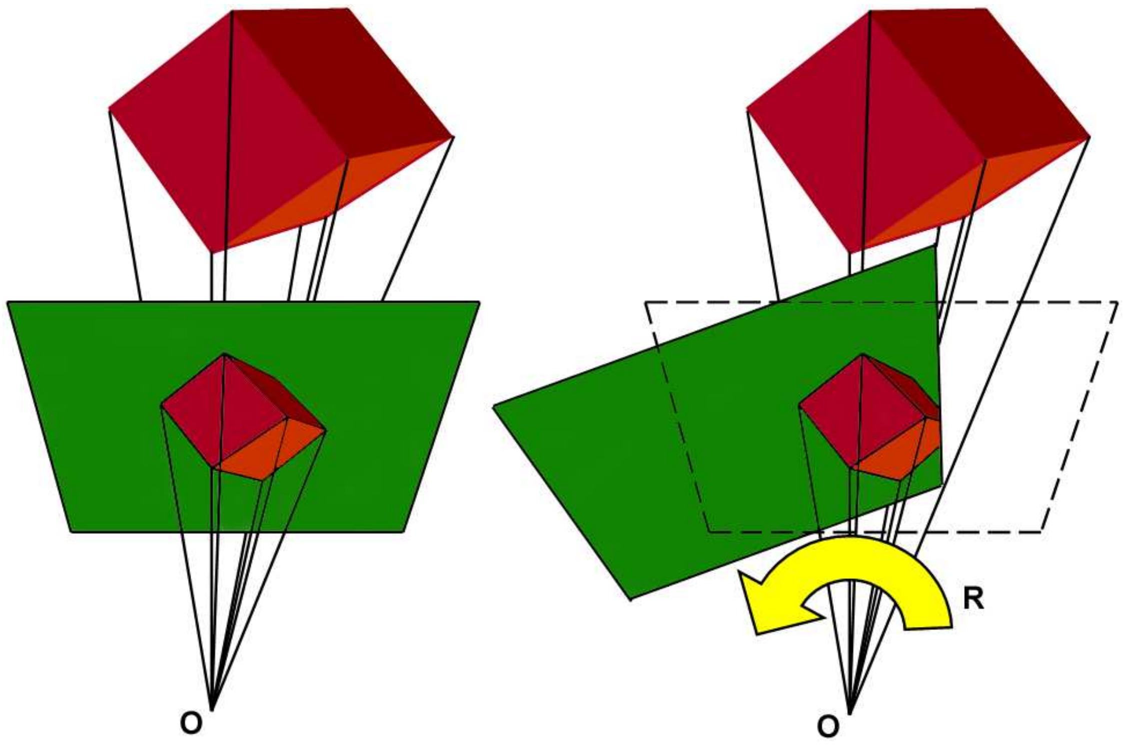

Figure 4.6: Pure rotation (R) about the camera center (O) is a special case of the general projective transformation.

4.2 Robust Estimation of F and H

We have argued that the velocity field between two images is constrained, and that given the matrices, **F** or **H**, we can restrict or predict the possible motions of a given point. We now turn to the robust estimation of **F** and **H**. The question remains as to how exactly we might identify corresponding points in order to form these linear constraints. As mentioned earlier in section 4.1.1, we use a Harris corner detector [54] to identify the points of interest in each image. After that, there are mainly two phrases involved in solving this correspondence or stereo matching problem: (1) Find an initial set of corner matches. (2) Remove corner mismatches from the initial set.

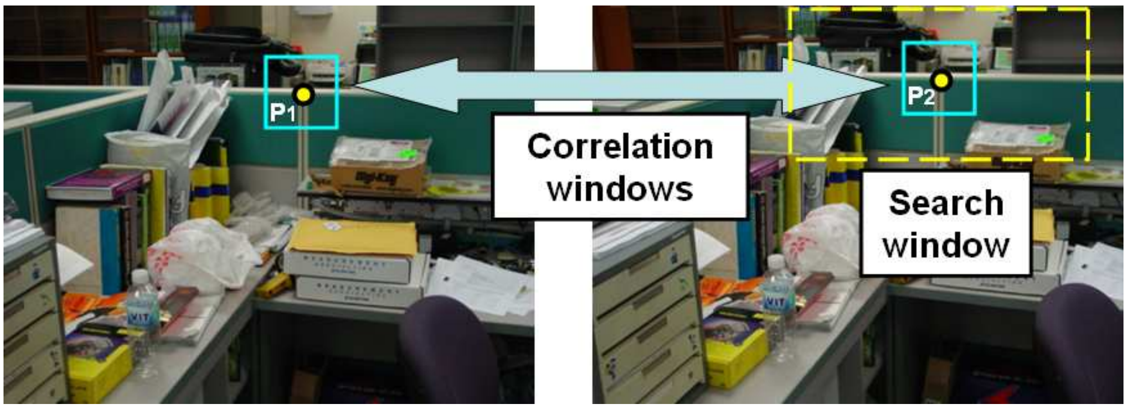

Figure 4.7: Initial Matching. Corner matches are assessed by performing cross-correlation of corner intensity neighborhoods. Based on assumption that the square neighborhoods of a corner match are similar with a high correlation score.

4.2.1 Initial Matching

The procedure implemented in this thesis work for obtaining an initial set of corner matches consists of the following steps:

Algorithm 4.1

1. First of all, we form a prediction for the movement of each corner from the first image to the second. The prediction may be based on previous estimates of camera motion or it might come from other tracking devices such as accelerometers. In this thesis work, the prediction was done by copying the previous estimates of camera motion.

2. This prediction constraint is used to restrict the search for potential corner matches. For a corner point $\mathbf{p}_1 = (x_1, y_1)$ in image 1, a small rectangular search area of size $(2d_x + 1) \times (2d_y + 1)$ centered about this point is defined in image 2 (Figure 4.7). This is equivalent to reducing the search area for a potential corner match in image 2 from the whole image to a given window. The search window reflects some *a priori* knowledge of the disparities between the matched corners.

3. Perform cross-correlation of corresponding pixels between a window of size $(2n+1) \times (2n+1)$ centered about $\mathbf{p}_1$ and similar windows centered about all detected corner points $\mathbf{p}_2 = (x_2, y_2)$ lying within the search window in image 2. The standard correlation score $\zeta(\mathbf{p}_1, \mathbf{p}_2)$ is defined as:

$$\zeta(\mathbf{p}_1, \mathbf{p}_2) = \frac{\sum_{i=-n}^{n} \sum_{j=-n}^{n} [I_1(x_1 + i, y_1 + j) - \overline{I_1(x_1, y_1)}][I_2(x_2 + i, y_2 + j) - \overline{I_2(x_2, y_2)}]}{(2n+1)^2 \sigma(I_1) \sigma(I_2)} \tag{4.7}$$

where $\overline{I_k(x_k, y_k)}$ is the average pixel intensity of a $(2n + 1) \times (2n + 1)$ neighborhood region about corner point (x_k, y_k) in image I_k $(k = 1, 2)$. $\sigma(I_k)$ is the standard deviation of the pixel intensities in the square neighborhood of corner point (x_k, y_k) in I_k. $\overline{I_k(x_k, y_k)}$ and $\sigma(I_k)$ are given by:

$$\begin{aligned}\overline{I_k(x_k, y_k)} &= \frac{\sum_{i=-n}^{n}\sum_{j=-n}^{n} I_k(x_k+i, y_k+j)}{(2n+1)^2} \\ \sigma(I_k) &= \sqrt{\frac{\sum_{i=-n}^{n}\sum_{j=-n}^{n} I_k^2(x_k+i, y_k+j)}{(2n+1)^2} - \overline{I_k(x_k, y_k)}}\end{aligned} \tag{4.8}$$

The score ranges from -1, for two correlation neighborhoods which are inverted, to 1, for two correlation neighborhoods which are identical. The pair of corner points with the highest correlation score is retained.

4. A constraint on the correlation score is then applied to select the most consistent matches: For the pair of corner points obtained in step 2, the correlation score must be higher than a given threshold to be considered a match candidate. If this requirement is satisfied, proceed to the next step, otherwise skip to step 5.

5. Repeat steps 1 and 2 but this time with the roles of the two images reversed. Specifically, for the match candidate $\mathbf{p}_2$ found in step 2, find the corner point in image 1 which gives the highest correlation score with $\mathbf{p}_2$. If the match candidate for $\mathbf{p}_2$ is again the initial $\mathbf{p}_1$, then this match will be validated; otherwise it will be rejected. This symmetric matching helps to reduce the probability of error matches.

6. Repeat steps 1-3 for the next detected corner point in image 1.

Thus, a set of initial corner matches is obtained based on constraints of proximity and similarity between corresponding local neighborhoods. The standard correlation score $\zeta(\mathbf{p}_1, \mathbf{p}_2)$ in Equation 4.7 is used to assess the degree of similarity i.e. the higher the score, the greater the similarity. Photometrically, $\zeta(\mathbf{p}_1, \mathbf{p}_2)$ is invariant to a linear transformation of image intensities – a constant addition and arbitrary scaling. Geometrically, $\zeta(\mathbf{p}_1, \mathbf{p}_2)$ is only invariant when the corner intensity neighborhoods are related by a simple translation.

4.2.2 Removing mismatches - RANSAC

We now have an initial set of matches (Figure 4.8). However, many of these are erroneous, and will render a brute-force least square solution ineffective. Here we adapted the RANdom SAmple Consensus (RANSAC) algorithm of Fischler and Bolles [36], which is a robust statistical procedure for fitting a model described by a set of parameters to data containing outliers i.e. data that are in gross disagreement with a postulated model. In the process of estimating the best solution for the model, a set of data points which are in agreement with this solution is also obtained.

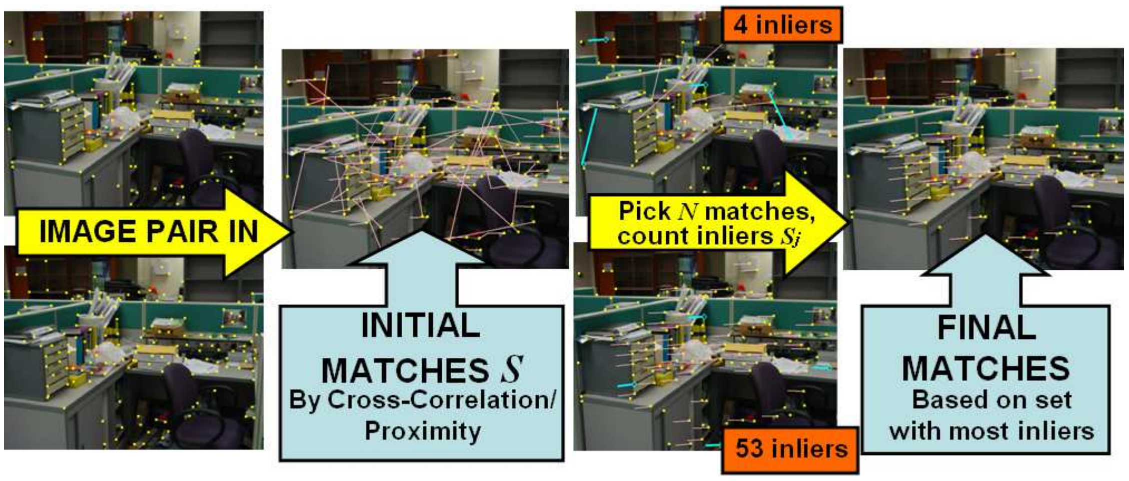
IMAGE PAIR IN
INITIAL MATCHES S
By Cross-Correlation/ Proximity
4 inliers
Pick N matches, count inliers S_j
53 inliers
FINAL MATCHES
Based on set with most inliers

Figure 4.8: Robust calculation of a homography between two images. Corner points are identified in the two images (yellow dots). We choose initial matches based on the similarity of the areas around these corners and on prior knowledge about the likely match direction (pink lines indicate corner vector to matched corner in other image). This initial set contains many incorrect matches. We pick N matches (blue lines) and calculate the associated homography. We then count the number of other matches that are in agreement (inliers are pink lines) and repeat this procedure. We choose the estimate with the most support and recalculate the homography using all of the inliers.

Given a set of initial corner matches (data) **S** containing mismatches (outliers), a minimal subset of the data (N matches) is randomly chosen to estimate a mathematical entity that describes the relative geometry between the two views (fitted model). The proposed solution of the relative image geometry is assessed by determining a set of corner matches $\mathbf{S}_j$ from **S** that is in agreement. The set $\mathbf{S}_j$ is the consensus set of the chosen subset sample and defines the inliers of **S**. To determine whether a corner match is consistent with the estimated model, the error of the corner match as described by a distance measure is compared to a user-defined threshold. The above procedure is repeated for a sufficient number of subset samples. The set $\mathbf{S}_j$ with the largest number of inliers is the set of final corner matches with outliers removed. The corresponding estimated model solution best describes the relative geometry between the two views.

In the following three sub-sections, we will discuss how exactly we calculate the transformation matrix for each set of matching points.

4.2.2.1 Fundamental Matrix

The fundamental matrix described in Section 4.1.2 provides a general representation of the relationship between corner matches in two views of the same scene. This relationship is the epipolar constraint in Equation 4.2. For a minimal number of eight corner matches, the fundamental matrix **F** can be computed by the normalized eight-point algorithm [55]. Given $n \geq 8$ corner matches $\{\mathbf{p}_i \leftrightarrow \mathbf{p}'_i\}_{i=1}^{n}$ with homogenous coordinates $\tilde{\mathbf{p}}_i = [x_i, y_i, 1]^T$ and $\tilde{\mathbf{p}}'_i = [x'_i, y'_i, 1]^T$, the method to compute **F** is as follows:

<u>Algorithm 4.2</u>

1. Normalize the coordinates in each image according to

$$\begin{aligned}\hat{\mathbf{p}}_i &= \mathbf{M}\tilde{\mathbf{p}}_i \\ \hat{\mathbf{p}}'_i &= \mathbf{M}'\tilde{\mathbf{p}}'_i\end{aligned} \tag{4.9}$$

where $\hat{\mathbf{p}}_i = [\hat{x}_i, \hat{y}_i, 1]^T$ and $\hat{\mathbf{p}}'_i = [\hat{x}'_i, \hat{y}'_i, 1]^T$. $\mathbf{M}$ and $\mathbf{M}'$ are transformations consisting of a translation and scaling of each image so that the centroid of the points is at the origin $(0, 0)$ and the average distance of the points from the origin is equal to $\sqrt{2}$. For example, $\mathbf{M}$ can be expressed as:

$$\begin{aligned}\mathbf{M} &= \begin{bmatrix} s & 0 & -s\bar{\mathbf{p}}(1) \\ 0 & s & -s\bar{\mathbf{p}}(2) \\ 0 & 0 & 1 \end{bmatrix} \\ \bar{\mathbf{p}} &= \frac{\sum_{i=1}^{n} \tilde{\mathbf{p}}_i}{n} \\ s &= \frac{n\sqrt{2}}{\sum_{i=1}^{n} \|\tilde{\mathbf{p}}_i - \bar{\mathbf{p}}\|}\end{aligned} \tag{4.10}$$

where $\| \bullet \|$ denotes vector norm and $\bar{\mathbf{p}}(i)$ is the *ith* component of $\bar{\mathbf{p}}$. The expression for $\mathbf{M}'$ is similar to Equation 4.10.

2. For the *ith* normalized corner match $\hat{\mathbf{p}}_i \leftrightarrow \hat{\mathbf{p}}'_i$, the epipolar constraint is expressed as follows:

$$\hat{\mathbf{p}}'^{T}_i \hat{\mathbf{F}} \hat{\mathbf{p}}_i = 0 \tag{4.11}$$

where $\hat{\mathbf{F}}$ is the normalized fundamental matrix. Each normalized corner match gives rise to one linear equation in the unknown entries of $\hat{\mathbf{F}}$. Specifically, Equation 4.11 can be expressed as a vector product:

$$\begin{bmatrix} \hat{x}'_i\hat{x}_i & \hat{x}'_i\hat{y}_i & \hat{x}'_i & \hat{y}'_i\hat{x}_i & \hat{y}'_i\hat{y}_i & \hat{y}'_i & \hat{x}_i & \hat{y}_i & 1 \end{bmatrix} \hat{\mathbf{f}} = 0$$

where $\hat{\mathbf{f}}$ is a 9×1 vector of the entries of $\hat{\mathbf{F}}$ in row-major order. Thus, for a set of n corner matches, construct a $n \times 9$ coefficient matrix $\mathbf{A}$ such that

$$\mathbf{A}\hat{\mathbf{f}} = \begin{bmatrix} \hat{x}'_1\hat{x}_1 & \hat{x}'_1\hat{y}_1 & \hat{x}'_1 & \hat{y}'_1\hat{x}_1 & \hat{y}'_1\hat{y}_1 & \hat{y}'_1 & \hat{x}_1 & \hat{y}_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{x}'_n\hat{x}_n & \hat{x}'_n\hat{y}_n & \hat{x}'_n & \hat{y}'_n\hat{x}_n & \hat{y}'_n\hat{y}_n & \hat{y}'_n & \hat{x}_n & \hat{y}_n & 1 \end{bmatrix} \hat{\mathbf{f}} = \mathbf{0}_{n\times 1} \quad (4.12)$$

The least-squares solution for $\hat{\mathbf{f}}$ is the singular vector corresponding to the smallest singular value of $\mathbf{A}$ i.e. the last column of $\mathbf{V}$ in the Singular Value Decomposition (SVD) $\mathbf{A} = \mathbf{UDV}^T$.

3. Denormalization. The fundamental matrix $\mathbf{F}$ corresponding to the original data $\{\mathbf{p}_i \leftrightarrow \mathbf{p}'_i\}_{i=1}^n$ is obtained as follows:

$$\mathbf{F} = \mathbf{M}'^T\hat{\mathbf{F}}\mathbf{M} \quad (4.13)$$

4.2.2.2 Homography

For the case in which the corner matches correspond to 3D points lying on a world plane or when the camera motion is pure rotation, the image motion between two views of the same scene is further constrained by Homography. For a corner match $\mathbf{p}_i \leftrightarrow \mathbf{p}'_i$, the following relationship stands:

$$\tilde{\mathbf{p}}'_i = \mathbf{H}\tilde{\mathbf{p}}_i \quad (4.14)$$

where $\mathbf{H}$ is the 3×3 homography matrix. As we mentioned before, this homography constraint is much stronger than the epipolar constraint since for a given $\mathbf{H}$, a prediction of where a point feature in one image will appear in the other image can be made. The homography can be computed from a minimal number of four corner matches. Given $n \geq 4$ corner matches $\{\mathbf{p}_i \leftrightarrow \mathbf{p}'_i\}_{i=1}^{n}$ with homogenous coordinates $\tilde{\mathbf{p}}_i = [x_i, y_i, 1]^T$ and $\tilde{\mathbf{p}}'_i = [x'_i, y'_i, 1]^T$, the method to compute $\mathbf{H}$ is as follows:

<u>Algorithm 4.3</u>

1. Normalize the corner points $\mathbf{p}_i$ and $\mathbf{p}'_i$ as in step 1 of Algorithm 4.2 to obtain $\hat{\mathbf{p}}_i = [\hat{x}_i, \hat{y}_i, 1]^T$ and $\hat{\mathbf{p}}'_i = [\hat{x}'_i, \hat{y}'_i, 1]^T$.

2. Equation 4.14 may be expressed in terms of the following vector cross product:

$$\begin{aligned}
\hat{\mathbf{p}}'_i \times \hat{\mathbf{H}}\hat{\mathbf{p}}_i &= \mathbf{0}_{3\times 1} \\
\begin{bmatrix} \hat{y}'_i \hat{\mathbf{h}}^{3T}\hat{\mathbf{p}}_i - \hat{\mathbf{h}}^{2T}\hat{\mathbf{p}}_i \\ \hat{\mathbf{h}}^{1T}\hat{\mathbf{p}}_i - \hat{x}'_i \hat{\mathbf{h}}^{3T}\hat{\mathbf{p}}_i \\ \hat{x}'_i \hat{\mathbf{h}}^{2T}\hat{\mathbf{p}}_i - \hat{y}'_i \hat{\mathbf{h}}^{1T}\hat{\mathbf{p}}_i \end{bmatrix} &= \mathbf{0}_{3\times 1} \\
\begin{bmatrix} \mathbf{0}_{1\times 3} & -\hat{\mathbf{p}}_i^T & \hat{y}'_i \hat{\mathbf{p}}_i^T \\ \hat{\mathbf{p}}_i^T & \mathbf{0}_{1\times 3} & -\hat{x}'_i \hat{\mathbf{p}}_i^T \\ -\hat{y}'_i \hat{\mathbf{p}}_i^T & \hat{x}'_i \hat{\mathbf{p}}_i^T & \mathbf{0}_{1\times 3} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{h}}^1 \\ \hat{\mathbf{h}}^2 \\ \hat{\mathbf{h}}^3 \end{bmatrix} &= \mathbf{0}_{3\times 1} \qquad (4.15)
\end{aligned}$$

where $\hat{\mathbf{h}}^{jT}$ denotes the *jth* row of the normalized homography $\hat{\mathbf{H}}$. Thus, each normalized corner match gives rise to three equations in the unknown entries of $\hat{\mathbf{H}}$ but only two of them are linearly independent. Since the third equation is the most complex one, we can omit it, and Equation 4.15 becomes:

$$\mathbf{A}_i \hat{\mathbf{h}} = \begin{bmatrix} 0 & 0 & 0 & -\hat{x}_i & -\hat{y}_i & -1 & \hat{y}'_i \hat{x}_i & \hat{y}'_i \hat{y}_i & \hat{y}'_i \\ \hat{x}_i & \hat{y}_i & 1 & 0 & 0 & 0 & \hat{x}'_i \hat{x}_i & \hat{x}'_i \hat{y}_i & \hat{x}'_i \end{bmatrix} \hat{\mathbf{h}} = \mathbf{0}_{2\times 1} \qquad (4.16)$$

where $\hat{\mathbf{h}}$ is a 9×1 vector of the entries of $\hat{\mathbf{H}}$ in row-major order . For a set of n corner matches, assemble the n 2×9 matrices $\mathbf{A}_i$ into a single $2n \times 9$

coefficient matrix $\mathbf{A}$. Thus,

$$\mathbf{A}\hat{\mathbf{h}} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \hat{\mathbf{h}} = \mathbf{0}_{2n\times 1} \tag{4.17}$$

The least-squares solution for $\hat{\mathbf{h}}$ is the singular vector corresponding to the smallest singular value of $\mathbf{A}$ i.e. the last column of $\mathbf{V}$ in the Singular Value Decomposition (SVD) $\mathbf{A} = \mathbf{UDV}^T$.

3. Denormalization. The homography matrix $\mathbf{H}$ corresponding to the original data $\{\mathbf{p}_i \leftrightarrow \mathbf{p}'_i\}_{i=1}^n$ is obtained as follows:

$$\mathbf{H} = \mathbf{M}'^{-1}\hat{\mathbf{H}}\mathbf{M} \tag{4.18}$$

where $\mathbf{M}$ and $\mathbf{M}'$ are the transformation matrices of step 1.

4.2.2.3 Affine Transform

A special case of homography known as the affine transform sometimes suffices to describe the 2D image motion. For a corner match $\mathbf{p}_i \leftrightarrow \mathbf{p}'_i$, the relationship is expressed as follows:

$$\tilde{\mathbf{p}}'_i = \mathbf{L}\tilde{\mathbf{p}}_i = \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{p}}_i \tag{4.19}$$

where $\mathbf{L}$ is the 3×3 affine transformation matrix. $\mathbf{L}$ has six unknowns compared to nine for the general homography $\mathbf{H}$. The affine transformation can be computed from a minimal number of three corner matches. Given $n \geq 3$ corner matches $\{\mathbf{p}_i \leftrightarrow \mathbf{p}'_i\}_{i=1}^n$ with homogenous coordinates $\tilde{\mathbf{p}}_i = [x_i, y_i, 1]^T$ and $\tilde{\mathbf{p}}'_i = [x'_i, y'_i, 1]^T$, the method to compute $\mathbf{L}$ is as follows:

Algorithm 4.4

1. From Equation 4.19, each corner match gives rise to two equations in the unknown entries of $\mathbf{L}$:

$$\begin{aligned} \begin{bmatrix} \mathbf{l}^{1T}\tilde{\mathbf{p}}_i \\ \mathbf{l}^{2T}\tilde{\mathbf{p}}_i \end{bmatrix} &= \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} \\ \begin{bmatrix} \tilde{\mathbf{p}}_i^T & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} & \tilde{\mathbf{p}}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{l}^1 \\ \mathbf{l}^2 \end{bmatrix} &= \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} \\ \mathbf{A}_i\mathbf{l} &= \mathbf{p}'_i \end{aligned} \tag{4.20}$$

where $\mathbf{l}^{jT}$ denotes the *jth* row of $\mathbf{L}$ and $\mathbf{l}$ is a 6×1 vector of the unknown entries in $\mathbf{L}$. For a set of n corner matches, assemble the n 2×6 matrices $\mathbf{A}_i$ into a single $2n \times 6$ matrix $\mathbf{A}$ and the n 2×1 vectors $\mathbf{p}'_i$ into a single $2n \times 1$ vector $\mathbf{p}'$. Thus,

$$\begin{aligned} \mathbf{A}\mathbf{l} &= \mathbf{p}' \\ \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \mathbf{l} &= \begin{bmatrix} \mathbf{p}'_1 \\ \vdots \\ \mathbf{p}'_n \end{bmatrix} \end{aligned} \tag{4.21}$$

2. From Equation 4.21, the least squares solution for $\mathbf{l}$ is as follows:

$$\mathbf{l} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{p}' \tag{4.22}$$

4.2.2.4 Distance measure and threshold

For the case of a fundamental matrix model, the distance measure is the symmetric epipolar distance of a matched corner pair from their respective predicted epipolar lines. The distance measure d_F is expressed as follows:

$$d_F = \left(\frac{1}{(\mathbf{F}\tilde{\mathbf{p}}_i)_1^2 + (\mathbf{F}\tilde{\mathbf{p}}_i)_2^2} + \frac{1}{(\mathbf{F}^{\mathrm{T}}\tilde{\mathbf{p}}_i')_1^2 + (\mathbf{F}^{\mathrm{T}}\tilde{\mathbf{p}}_i')_2^2} \right) (\tilde{\mathbf{p}}_i'^{\mathrm{T}}\mathbf{F}\tilde{\mathbf{p}}_i)^2 \tag{4.23}$$

where $\mathbf{p}_i \leftrightarrow \mathbf{p}_i'$ is a corner match from the initial set of matches $\mathbf{S}$ and $\mathbf{F}$ is an estimate of the fundamental matrix from a RANSAC sample of eight randomly chosen corner matches.

For the case of a homography model, the following distance measure consisting of the symmetric reprojection errors is defined:

$$d_H = d(\mathbf{p}_i', \mathbf{H}\tilde{\mathbf{p}}_i)^2 + d(\mathbf{p}_i, \mathbf{H}^{-1}\tilde{\mathbf{p}}_i')^2 \tag{4.24}$$

where $d(\mathbf{a}, \mathbf{b})$ is simply the Euclidean distance between measured corner position $\mathbf{a}$ and predicted position $\mathbf{b}$. $\mathbf{H}$ is an estimate of the homography from a RANSAC sample of four randomly chosen corner matches.

The distances d_F and d_H measure how closely a matched corner pair satisfies the proposed epipolar geometry and homography relationship respectively. If the calculated distance is less than a certain threshold d_t, the corner match is deemed an inlier to the model solution, otherwise it is an outlier. Under the assumption that the measurement error is Gaussian with zero mean and standard deviation σ, a value for d_t can be determined in a statistical manner. Details of the computation are provided in [56]. For a 95% probability that a corner match is correctly classified as an inlier, the values of d_t for the fundamental matrix and homography models are in Table 4.1. Considering that the 2D positions of the corners are measured up

Model	d_t
Fundamental matrix	$3.84\sigma^2$
Homography	$5.99\sigma^2$

Table 4.1: Distance thresholds.

to subpixel accuracy (Section 4.1.1), a σ value of 0.5 is observed to work well in practice.

4.2.2.5 Number of samples

The RANSAC procedure is repeated for N_s samples, where each sample consists of N randomly chosen corner matches from the initial set of matches $\mathbf{S}$. For a fundamental matrix model, $N = 8$ and for a homography model, $N = 4$. N_s is chosen sufficiently high to ensure with a probability p that at least one of the random samples is free from outliers. Suppose w is the probability that any selected corner match is an inlier, thus $\epsilon = 1 - w$ is the probability that it is an outlier. Then at least N_s selections, each of N corner matches, are required. Thus,

$$\begin{aligned} \left(1 - w^N\right)^{N_s} &= 1 - p \\ N_s &= \frac{\log(1-p)}{\log\left(1 - (1-\epsilon)^N\right)} \end{aligned} \tag{4.25}$$

For the fundamental matrix and homography models, examples of N_s for $p = 0.99$ and different values of ϵ are tabulated in Table 4.2.

Sample size	Proportion of outliers ϵ				
N	0.1	0.2	0.3	0.4	0.5
4	5	9	17	34	72
8	9	26	78	272	1177

Table 4.2: Number of RANSAC samples.

In this table, the first row shows the proportion of the outliers, which ranges from 0.1 to 0.5. The left column shows the sample size for estimating homography and fundamental matrix, which are 4 and 8 respectively. And the number in the right bottom box is the number of required samples needed (it is also the number of RANSAC loop needed) before obtaining an accurate homography or fundamental matrix.

As we can see from the table, when the proportion of outliers increases, there is a severe increase in the computational cost incurred by the increase in the number of required samples, N_s. Thus, it is important to obtain a high quality set of initial matches with a high proportion of correct matches so that the RANSAC procedure is not over-burdened and the probability of any remaining false matches is also reduced. For the narrow baseline case, Algorithm 4.1 typically generate sets of initial matches that are already $> 70\%$ correct.

Thus, given a set of initial corner matches $\mathbf{S}$ obtained by Algorithm 4.1, the RANSAC procedure based on a fundamental matrix model is summarized as follows:

Algorithm 4.5

1. Repeat for N_s samples, where N_s is determined from Equation 4.25 for $N = 8$ and a user-defined ϵ.

   - Select a random sample of $N = 8$ corner matches and compute the fundamental matrix $\mathbf{F}$ by Algorithm 4.2.
   - Calculate the distance d_F (Equation 4.23) for each corner match in $\mathbf{S}$.
   - Compute the number of inliers consistent with $\mathbf{F}$ by the number of corner matches for which $d_F < d_t$ pixels. d_t is determined from Table 4.1 and let this set of inliers be $\mathbf{S}_j$ for $j = 1, \ldots, N_s$.

2. Retain the set $\mathbf{S}_j$ with the largest number of inliers. This is the set of final corner matches with outliers removed.

For a homography model, Algorithm 4.5 is modified accordingly by setting $N = 4$. The homography and distance d_H are computed using Algorithm 4.3 and Equation 4.24 respectively.

Alternative robust statistical methods that can be used include the Least Median of Squares, case deletion and M-estimation algorithms. Torr and Murray [57] provided a review and comparison of the various methods.

4.3 Estimation of Relative Camera Motion

After calculation of either of above procedures, we are left with a correct set of two-view corner matches between $\mathbf{V}_k$ and reference image $\mathbf{V}$, $\{\mathbf{p}_{kj} \leftrightarrow \mathbf{p}_j\}_{j=1}^{\mathrm{N}}$.

To estimate the camera pose or motion $\mathbf{T}_k$ of $\mathbf{V}_k$ relative to $\mathbf{V}$, the standard least-squares method is adopted in this project, which minimizes deviations from the two-view constraints as a function of the motion parameters of $\mathbf{T}_k$. A linear algorithm based on 3D-2D point correspondences provides the initial estimate of $\mathbf{T}_k$ for the non-linear minimization.

4.3.1 Motion Parameterization

The relative camera motion $\mathbf{T}_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \end{bmatrix}$ is a 6DOF measurement and can be parameterized by a 6×1 motion vector $\mathbf{x}_k$:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{r}_k \\ \mathbf{t}_k \end{bmatrix} \tag{4.26}$$

where the rotation is represented by a 3×1 rotation vector $\mathbf{r}_k$. This is also known as the axis-angle representation where $\|\mathbf{r}_k\|$ is the rotation angle and $\mathbf{r}_k/\|\mathbf{r}_k\|$ is the unit norm rotation axis. The $\mathbf{x}_k$ parameterization is particularly useful as the non-linear minimization can now be carried out over a 6D parameter space corresponding to the 6 unknowns of $\mathbf{x}_k$ instead the original 12D space due to the 12 unknowns in $\mathbf{T}_k$. This results in considerable savings in computational cost for the minimization procedure. But most importantly, more degrees of freedom are involved with a higher dimensional space, resulting in a greater risk of a non-unique solution. The conversions between the two rotation representations [58] are summarized in following two subsections.

Axis-angle to rotation matrix

A rotation vector $\mathbf{r}$ is converted to its matrix representation $\mathbf{R}$ by the Rodrigue's formula:

$$\begin{aligned} \mathbf{R} &= \mathbf{I}_{3\times3} + f(\theta)[\mathbf{r}]_\times + g(\theta)([\mathbf{r}]_\times)^2 \\ \theta &= \|\mathbf{r}\| \\ f(\theta) &= \frac{\sin\theta}{\theta} \\ g(\theta) &= \frac{1-\cos\theta}{\theta^2} \end{aligned} \tag{4.27}$$

where $\mathbf{I}_{3\times3}$ is the 3×3 identity matrix and $[\mathbf{r}]_\times$ is a skew-symmetric matrix of the rotation vector $\mathbf{r}$, which is expressed as in Equation 4.4.

Rotation matrix to axis-angle

The solution to the inverse problem is as follows:

$$\begin{aligned}
\mathbf{r} &= \theta \frac{\mathbf{d}}{\|\mathbf{d}\|} \\
\theta &= \cos^{-1}\left[\frac{trace(\mathbf{R}) - 1}{2}\right] \\
\mathbf{d} &= \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \\
\mathbf{R} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}
\end{aligned} \qquad (4.28)$$

where $trace(\mathbf{R})$ is the sum of the diagonal terms of $\mathbf{R}$ i.e. $r_{11} + r_{22} + r_{33}$.

4.3.2 Least-Squares Estimation

From the two-view corner matches $\{\mathbf{p}_i \leftrightarrow \mathbf{p}_{ki}\}_{i=1}^{\mathrm{N}}$ and their reconstructed 3D coordinates $\{\mathbf{P}_i\}_{i=1}^{\mathrm{N}}$, a set of 3D-2D correspondences $\{\mathbf{P}_i \leftrightarrow \mathbf{p}_{ki}\}_{i=1}^{\mathrm{N}}$ between the 3D points and their projections in $\mathbf{V}_k$ can be set up. Similar to Equation 2.1, the perspective relationship between a 3D-2D point correspondence $\mathbf{P}_i \leftrightarrow \mathbf{p}_{ki}$ is as follows:

$$s\tilde{\mathbf{p}}_{ki} = \mathbf{K}\mathbf{T}_k\tilde{\mathbf{P}}_i \qquad (4.29)$$

where $\mathbf{K}$ is the intrinsic parameter matrix obtained in Section 2.4.2. $\mathbf{T}_k$ has the following form:

$$\mathbf{T}_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{1T} & t_x \\ \mathbf{r}^{2T} & t_y \\ \mathbf{r}^{3T} & t_z \end{bmatrix} = \begin{bmatrix} \mathbf{t}^{1T} \\ \mathbf{t}^{2T} \\ \mathbf{t}^{3T} \end{bmatrix}$$

where $\mathbf{r}^{jT}$ and $\mathbf{t}^{jT}$ is the *jth* row of the rotation matrix $\mathbf{R}_k$ and $\mathbf{T}_k$ respectively. The translation vector $\mathbf{t}_k = [t_x, t_y, t_z]^T$. $\mathbf{R}_k$ must satisfy a set of orthonormality constraints:

- $\mathbf{R}_k^T\mathbf{R}_k = \mathbf{I}_{3\times3}\ \|\mathbf{r}^{jT}\| = 1$

From Equation 4.29 and the orthonormality constraints, an analytical approach is developed to estimate the rotation and translation parameters of $\mathbf{T}_k$. This estimate of $\mathbf{T}_k$ is then converted to a motion vector representation. The method is described as follows:

Algorithm 4.6

1. Express $\mathbf{p}_{ki}$ in terms of camera coordinates:

$$\hat{\mathbf{p}}_{ki} = \mathbf{K}^{-1}\tilde{\mathbf{p}}_{ki} = [\hat{x}_i, \hat{y}_i, 1]^T \tag{4.30}$$

2. From step 1 and eliminating the arbitrary scaling factor s from Equation 4.29,

$$\hat{x}_i = \frac{\mathbf{t}^{1T}\tilde{\mathbf{P}}_i}{\mathbf{t}^{3T}\tilde{\mathbf{P}}_i} = \frac{X_c}{Z_c} \tag{4.31}$$

$$\hat{y}_i = \frac{\mathbf{t}^{2T}\tilde{\mathbf{P}}_i}{\mathbf{t}^{3T}\tilde{\mathbf{P}}_i} = \frac{Y_c}{Z_c} \tag{4.32}$$

where (X_c, Y_c, Z_c) are the coordinates of the 3D point $\mathbf{P}_i$ relative to the camera coordinate system of $\mathbf{V}_k$. Since Equations 4.31 and 4.32 have the same denominator, the following equation can be formed:

$$\begin{aligned} \begin{bmatrix} \hat{y}_i\tilde{\mathbf{P}}_i^T & -\hat{x}_i\tilde{\mathbf{P}}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{t}^1 \\ \mathbf{t}^2 \end{bmatrix} &= 0 \\ \text{or}: \mathbf{A}_i\mathbf{v} &= 0 \end{aligned} \tag{4.33}$$

where $\mathbf{A}_i$ is a 1×8 row vector and $\mathbf{v}$ is a 8×1 column vector of the unknown entries in the first two rows of $\mathbf{T}_k$. Thus, for a set of N 3D-2D point matches, construct a $\mathrm{N} \times 8$ coefficient matrix $\mathbf{A}$ such that

$$\mathbf{Av} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_\mathrm{N} \end{bmatrix} \mathbf{v} = \mathbf{0}_{\mathrm{N}\times 1} \tag{4.34}$$

The least-squares solution for $\mathbf{v}$ is the singular vector corresponding to the smallest singular value of $\mathbf{A}$ i.e. the last column of $\mathbf{V}$ in the Singular Value Decomposition (SVD) $\mathbf{A} = \mathbf{UDV}^T$. The solution $\bar{\mathbf{v}}$ is only unique up to an unknown scale factor γ:

$$\bar{\mathbf{v}} = \gamma\mathbf{v} = \gamma[\mathbf{r}^{1T}, t_x, \mathbf{r}^{2T}, t_y]^T \tag{4.35}$$

3. Determine the unknown scaling factor γ. Since $\|\mathbf{r}^{1T}\| = \|\mathbf{r}^{2T}\| = 1$, from the first 3 components of $\bar{\mathbf{v}}$ in Equation 4.35,

$$\sqrt{\bar{v}_1^2 + \bar{v}_2^2 + \bar{v}_3^2} = |\gamma|\|\mathbf{r}^{1T}\| = |\gamma| \tag{4.36}$$

where $\bar{v}_j$ is the *jth* component of vector $\bar{\mathbf{v}}$. Thus, the first two rows of the rotation matrix and the first two components of the translation vector are estimated up to an unknown common sign:

$$\begin{aligned}
\hat{\mathbf{r}}^{1T} &= \frac{1}{|\gamma|}[\bar{v}_1, \bar{v}_2, \bar{v}_3] \\
\hat{\mathbf{r}}^{2T} &= \frac{1}{|\gamma|}[\bar{v}_5, \bar{v}_6, \bar{v}_7] \\
t_x &= \frac{1}{|\gamma|}\bar{v}_4 \\
t_y &= \frac{1}{|\gamma|}\bar{v}_8 \qquad (4.37)
\end{aligned}$$

4. Determine the rotation matrix $\mathbf{R}_k$. The third row of the rotation matrix is obtained as the vector product of the first two estimated rows. Thus, the estimated rotation matrix $\hat{\mathbf{R}}_k$ is as follows:

$$\begin{aligned}
\hat{\mathbf{R}}_k &= \begin{bmatrix} \hat{\mathbf{r}}^{1T} \\ \hat{\mathbf{r}}^{2T} \\ \hat{\mathbf{r}}^{3T} \end{bmatrix} \\
\hat{\mathbf{r}}^{3T} &= \hat{\mathbf{r}}^{1T} \times \hat{\mathbf{r}}^{2T} \qquad (4.38)
\end{aligned}$$

$\hat{\mathbf{R}}_k$ is in general not orthogonal i.e. $\hat{\mathbf{R}}_k^T\hat{\mathbf{R}}_k \neq \mathbf{I}_{3\times3}$. To enforce the orthogonality constraint, perform SVD on $\hat{\mathbf{R}}_k$ such that $\hat{\mathbf{R}}_k = \mathbf{UDV}^T$, then replace $\mathbf{D}$ with the 3×3 identity matrix $\mathbf{I}_{3\times3}$ so that the resulting rotation matrix $\mathbf{R}_k = \mathbf{UI}_{3\times3}\mathbf{V}^T$ is orthogonal.

5. Determine the unknown sign of γ. The depth Z_c of the 3D point $\mathbf{P}_i$ relative to the camera coordinate system of $\mathbf{V}_k$ must be positive since $\mathbf{P}_i$ must be

infront of the camera. From Equation 4.31, this means that

$$\frac{\mathbf{t}^{1T}\tilde{\mathbf{P}}_i}{\hat{x}_i} > 0 \tag{4.39}$$

The parameters of $\mathbf{t}^{1T} = [\mathbf{r}^{1T}, t_x]$ are obtained in steps 3 and 4. It is sufficient to do this check for one of the 3D-2D point correspondences. If the above constraint is not satisfied, reverse the signs of the first two rows of $\mathbf{R}_k$ and the first two components (t_x, t_y) of the translation vector.

6. Determine the third component t_z of the translation vector. Equation 4.31 can be rearranged into the following form:

$$\begin{aligned} \hat{x}_i t_z &= \mathbf{t}^{1T}\tilde{\mathbf{P}}_i - \hat{x}_i \mathbf{r}^{3T}\mathbf{P}_i \\ \hat{x}_i t_z &= b_i \end{aligned} \tag{4.40}$$

where the parameters of $\mathbf{t}^{1T}$ and $\mathbf{r}^{3T}$ are obtained from steps 3-5. Thus for a set of N 3D-2D point matches, a linear system of equations can be formed:

$$\begin{aligned} \hat{\mathbf{x}} t_z &= \mathbf{b} \\ \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_{\mathrm{N}} \end{bmatrix} t_z &= \begin{bmatrix} b_1 \\ \vdots \\ b_{\mathrm{N}} \end{bmatrix} \end{aligned} \tag{4.41}$$

The least squares solution for t_z is as follows:

$$t_z = (\hat{\mathbf{x}}^T\hat{\mathbf{x}})^{-1}\hat{\mathbf{x}}^T\mathbf{b} \tag{4.42}$$

7. Convert $\mathbf{T}_k$ to the motion vector $\mathbf{x}_k$ as described in Section 4.3.1.

The linear estimate of $\mathbf{x}_k$ obtained in Algorithm 4.6 is refined by minimizing a non-linear criterion function ε_s of the two-view costs for the case of homography, which is the sum of squared Euclidean distance between measured corner position $\mathbf{p}_{ki}$ and predicted position $\mathbf{p}'_{ki}$ in $\mathbf{V}_k$ for each 3D-2D point correspondence $\mathbf{P}_i \leftrightarrow \mathbf{p}_{ki}$. This standard least-squares formulation is as follows:

$$\min_{\mathbf{x}_k} \varepsilon_s(\mathbf{x}_k) = \mathbf{z}_k^T \mathbf{z}_k \tag{4.43}$$

where $\mathbf{z}_k$ is a (N $\times$ 1) vector of measurement errors,

$$\mathbf{z}_k = \begin{bmatrix} \|\mathbf{p}_{k1} - \mathbf{p}'_{k1}\| \\ \|\mathbf{p}_{k2} - \mathbf{p}'_{k2}\| \\ \|\mathbf{p}_{k3} - \mathbf{p}'_{k3}\| \\ \vdots \\ \|\mathbf{p}_{k\mathrm{N}} - \mathbf{p}'_{k\mathrm{N}}\| \end{bmatrix} \tag{4.44}$$

The criterion function ε_s is minimized over the 6 motion parameters of $\mathbf{x}_k$ by an iterative minimization method: Given an initial estimate of the motion vector $\mathbf{x}_k$ from Algorithm 4.6 and the corresponding value of ε_s, the minimization method seeks a new estimate of $\mathbf{x}_k$ which results in a smaller value for ε_s. The search direction is based on the gradient of ε_s at the estimate of $\mathbf{x}_k$ in the previous iteration. This procedure is repeated until a minimum value of value of ε_s is reached and the corresponding $\mathbf{x}_k$ is then the optimal motion estimate. Common minimization techniques include the Newton's and Levenberg-Marquardt (LM) methods. These techniques mainly differ in the way the function gradient is used to update the current parameter vector estimate. For this application, the LM minimization method is chosen due to its fast convergence. The Numerical Recipes [59] provides a good implementation of the LM method.

Chapter 5

Results of the Natural Feature Applications

We are able to directly use the homography and fundamental matrix to estimate the flow field for 2-D feature points in the image, which means a real-time 2-D geographical labeling is achievable. However, these mathematical entities can also be used to form estimates of the full 3-D camera motion as described in the previous chapter. An example of this can be found in the short video clip Cube.avi in the attached CD-ROM. Two sample frames are shown in Figure 5.1.

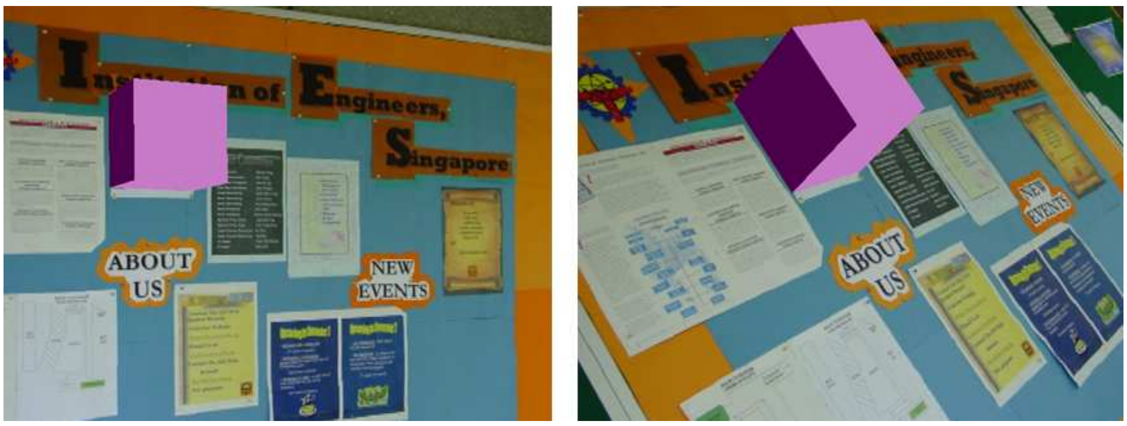

Figure 5.1: Superimpose 3D graphical content onto a real notice board.

For more general scenes, the fundamental matrix encompasses information about camera movement, but there are a number of problematical aspects to calculating 3D camera motion in real time, including the ambiguity of scale, and the fact that the calculation is unstable when the translational component is small or the scene structure is not sufficiently general. In these cases a homography may suffice. In this project work, we developed two natural feature AR systems based on homography constraints.

5.1 Geographic Labelling/Overlaying Application

The first application that I have developed in this project work is a wide-scale geographical labeling/overlaying system. We store a number of key frames (see Figure 5.2) of the geographical features in question. For each key frame we store the positions of the corners in the image, and the local image intensities around them. At a given position in space, we may more generally sample more visual directions than is possible with a single image by applying mosaicing techniques. The system attempts to find the velocity map between the current frame and the stored frame. The positions of the geographical annotations are then mapped from the stored frame to the current frame and drawn into the field. This method has a major advantage over previous systems [37] in that it always matches back to the reference frame rather than repeatedly updating the position of the labels based on the previous frame. Hence, errors do not accrue over time. Moreover, because the matching procedure is robust, it will not fail if part of the image has changed in the time between initial capture and the present.

The user with a wearable computer (Pentium III 1Ghz) in his jacket and a HMD on his head (Figure 5.3) stands in front of different buildings in the National University of Singapore campus. From there, he would be able to see a virtual

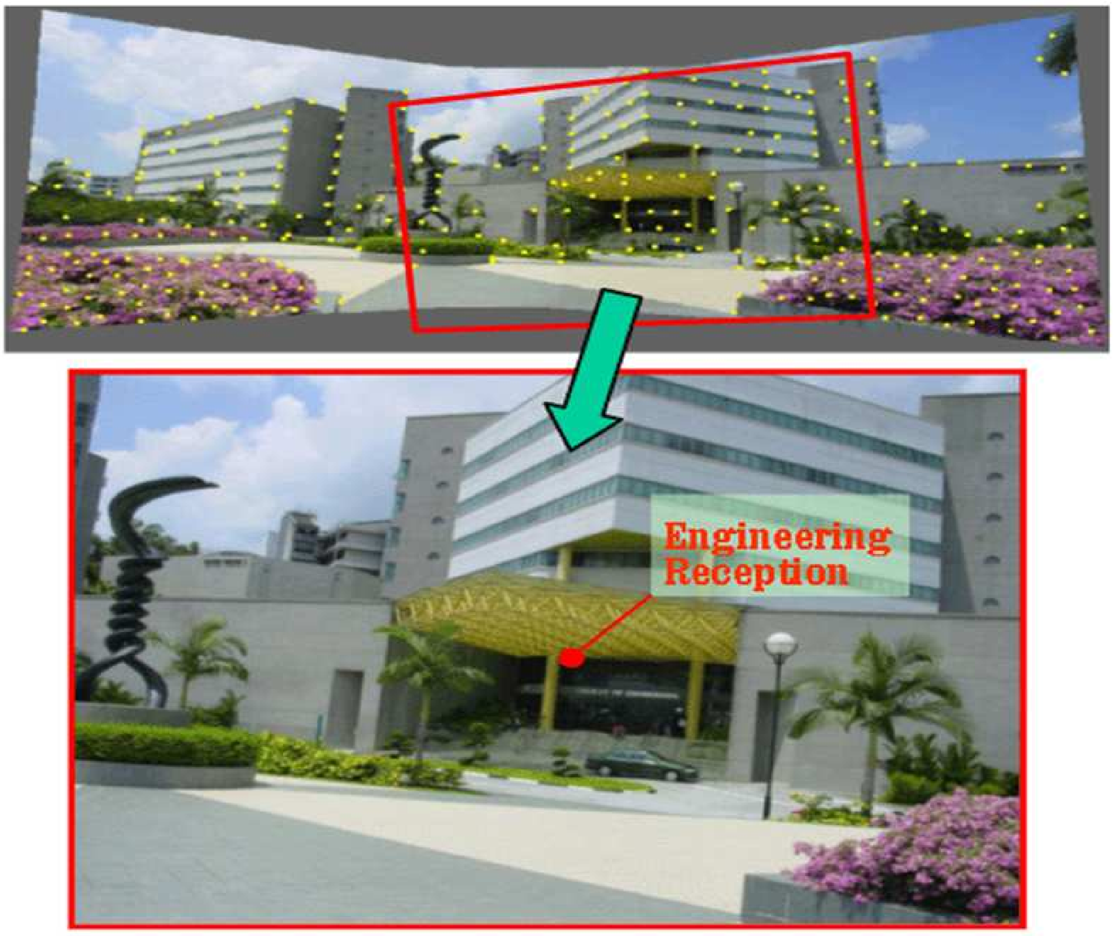

Figure 5.2: Information for geographical annotation can be stored in the form of corner points and surrounding regions and their directions in space. These points may be stored at a given point across a wide range of angles (displayed top as a mosaic). The input frame (bottom) can be compared to this stored representation to establish the position of the label.

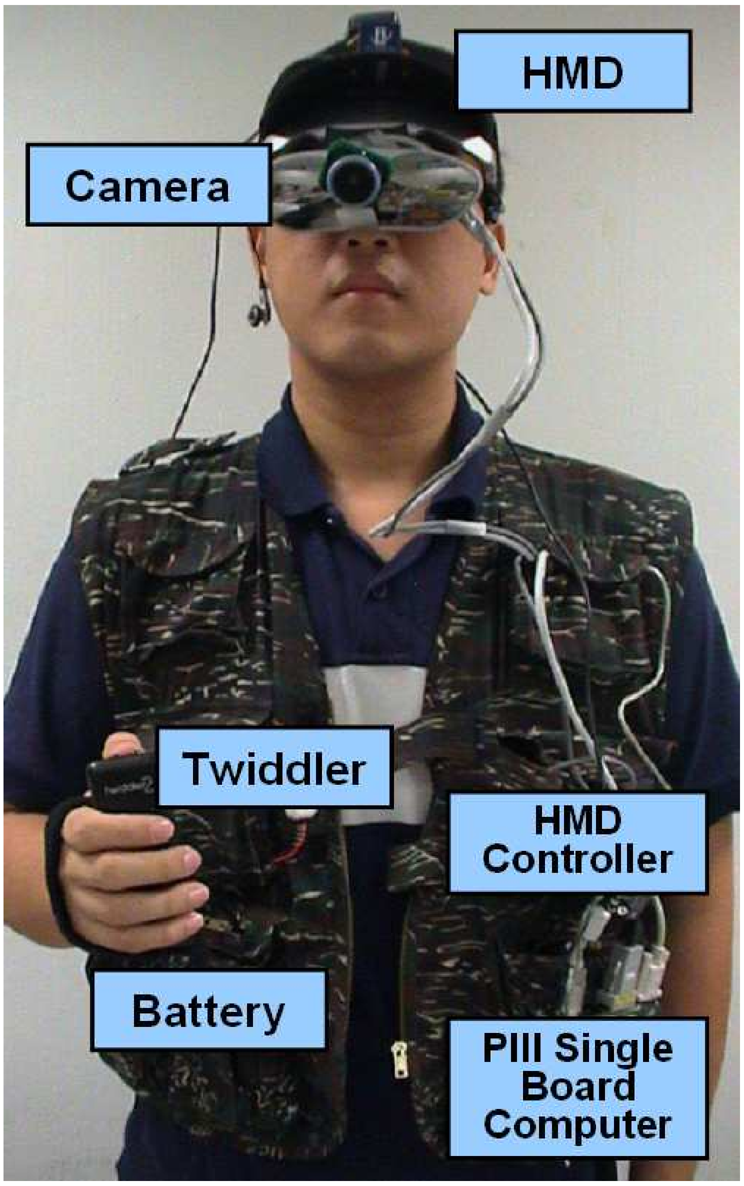

Figure 5.3: Wearable computer used in the geographical labeling/overlaying system.

text label or a 3D virtual object (a virtual 3D model of the building for example) being attached to the building in front of him. When he moves his head around, the label is always sticking to one particular part of building. Because the motion of the user's head can be regarded as a pure rotation when we compare it with the distance between him and the building in front, we can use the estimation of the homography to integrate fully three-dimensional objects into the scene.

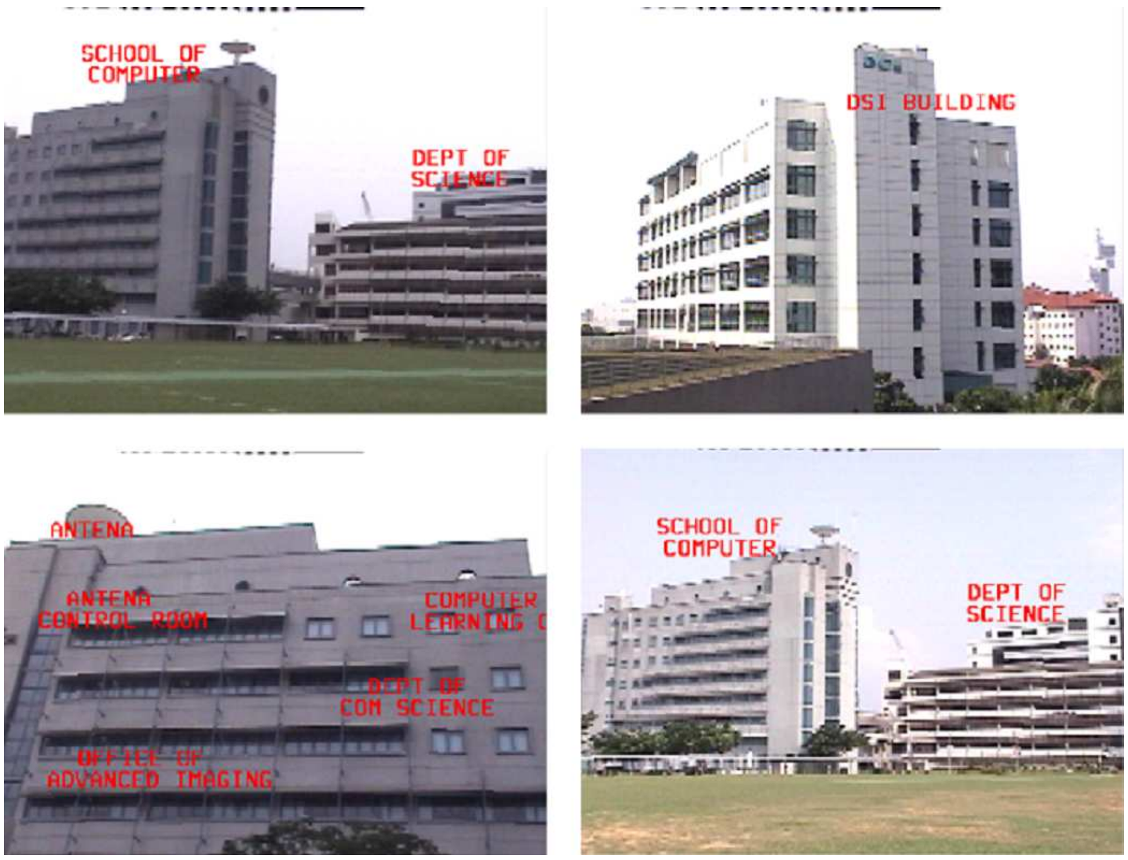

Figure 5.4: The geographical labelling of different buildings in the National University of Singapore Kent Ridge campus.

Reliable calculation of the fundamental matrix is currently rather slow for real time wearable computing applications, and only operates at <8Hz on our wearable platform which is based on a 1GHz 786 single board computer from Inside Technologies, and a Sony Glasstron display. However another wearable system we have developed based on a Dell 2.4 Ghz Pentium IV Laptop can calculate homography matrices at 25Hz between 320×240 pixel grayscale images.

The video file TextLabel.avi in the attached CD-ROM shows the 2D text labelling result of our natural feature tracking system, and Figure 5.4 shows some screen shots of it. The experiment was carried out in front of different buildings in the Kent Ridge campus. As shown in the video, I have tested to walk in all 8 main directions (north, south, east, west, northeast, northwest, southeast, southwest), and the system still remains very robust.

In the video TextLabel.avi, the wording *School of Computer* disappears the moment these words touch the left margin. This is because the text was drawn using OpenGL, and the reference point of these wording is at the left upper point of the wording area. So when the reference point is out of the image (for example, the *S* for *School of Computer* is out of the image area), OpenGL will fail to draw the words. This problem can be solved by shifting the reference point to the center of the wording area.

The second video file BuildingOverlay.avi in the attached CD-ROM shows the 3D virtual building overlaying result of the system. Some screen shots are shown in Figure 5.5. Figure 5.5-[a] shows the building we are shooting at. The pictures in the left column are the results when we augmented the virtual building onto the real one. By pressing a predefined key on the Twiddler [60], the user can also decide the transparency level of this virtual building as shown in the three pictures in the right column.

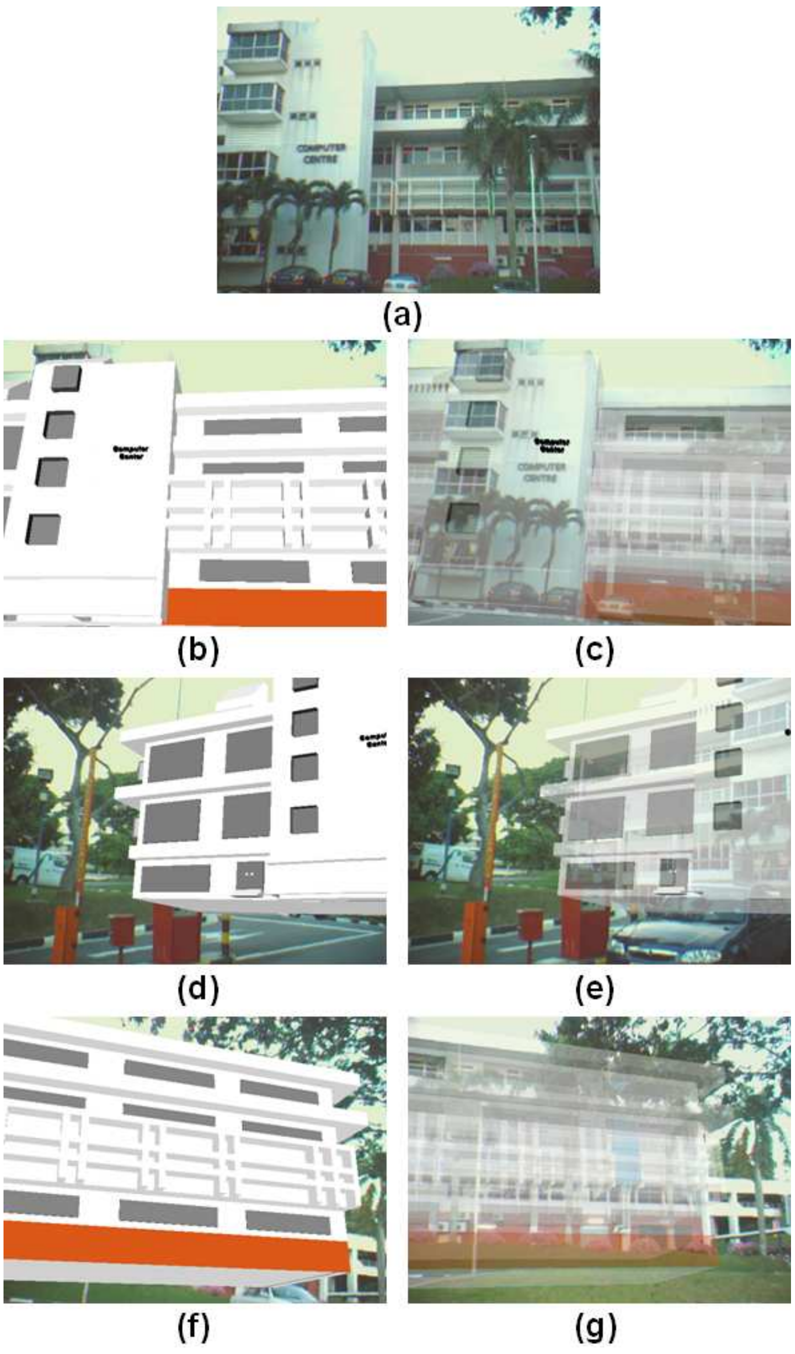

Figure 5.5: The 3D geographical overlaying of the Computer Center building: The three images in the left column show the situation when the system augments a fully solid on top of the original Computer Center. The three images in the right column show the situation when the user prefers to see a half transparent building structure by using keys on the Twiddler.

5.2 Room Decorative Application

In addition to the outdoor application, an indoor room decorative application was also built in this thesis work.

The objective of this application is to put a virtual ceiling light in the real room. After the program starts, the user will see a red cross through his HMD. That's where he's going to place the ceiling light. After he aims the red cross properly to one point on the ceiling, by pressing a predefined key on the Twiddler, he will be able to add the virtual ceiling light to that point in the scene as shown in Figure 5.6. A short video clip IndoorLight.avi of the result of this application can be found in the attached CD-ROM. By pressing some other predefined keys on the Twiddler, the user can even load different virtual lights into the scene, or even rotate and move the light around.

Same as the geographic labelling system, the room decorative application was also built on the estimation of the Homographies between the incoming frames and the pre-stored key frames. The structure of the room is obviously not planar. This is the same for the various buildings in the geographic labelling/overlaying application. Because of this, we expect the observer only rotate the camera when he looks around with the HMD.

However, during the experiments, we found out some translations were still acceptable for a accurate tracking in both applications, especially in case of the outdoor geographical labeling/overlaing as the buildings are normally quite a distance away from the observer. Even when the observer walked within a 2 meter by 2 meter region, the homography still remained as a very good estimation of the motion field as shown in the videos. Details of the performances of these systems are discussed in Chapter 6.

In the video clip IndoorLight.avi, the virtual lamp sometimes may drift a bit from its original position. This is because, in this application, due to the lighting condition in the room and the speed of camera rotation, homography may fail to be accurately estimated t certain moment. When that happens, the reference frame will be automatically replaced by the current image frame in order to increase the probability of successful estimation of the next homography value. As a result, errors may accumulate, and the virtual lamp will seem to be drifting a bit.

Also, there is only one camera used in this project for tracking, so the depth can not be accurately detected by the system. And the occlusion is very difficult to be implemented here either, because of the realtime requirement. That is why only rotation and small translation of the camera are recommended in this application, and the virtual lamp may not seem to be very *immersive* into the real scene from the video.

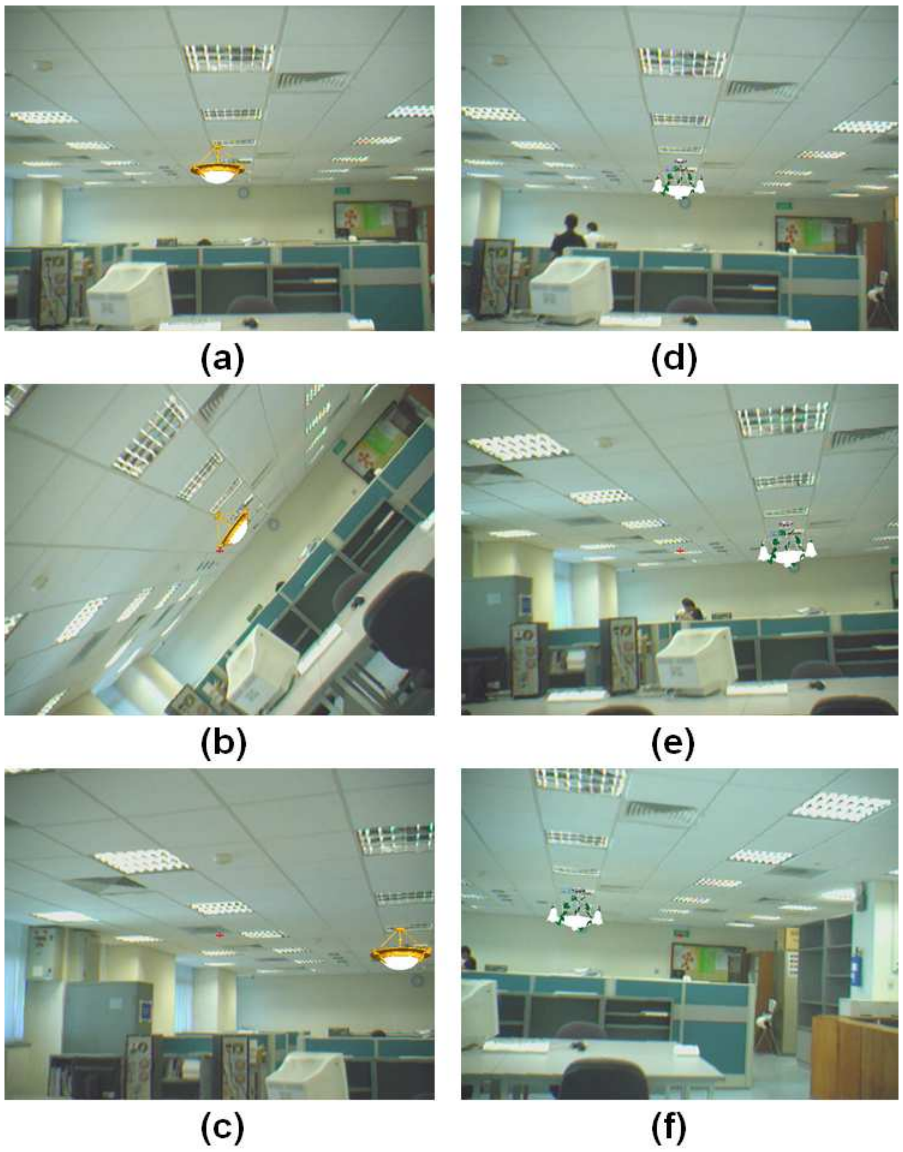

Figure 5.6: The Room Decoration Application.

5.3 Pacman Game Application

In addition to the geographical labeling/overlaying system and room decorative application, we also applied this robust natural feature tracking algorithm to a real-time handhold 3D Pacman game.

Pacman is always one of the popular PC games. The objective of this application is to develop a 3 dimensional Pacman game in user's hands based on the AR technologies as well as the proposed natural feature tracking algorithm. First, we designed a colorful piece of paper (Figure 5.7), which has lots of feature variations. We pasted this piece of paper on a A4 size hard board, and use it as the reference frame for estimating the position of the 3D Pacman maze. As discussed before, the selection of the initial matches is generally based on the similarity of small regions surrounding the feature corners, and on *a prior* estimate of the image velocity. This tracking information is used to define the displacement for both the three dimensional maze and the virtual characters in the maze.

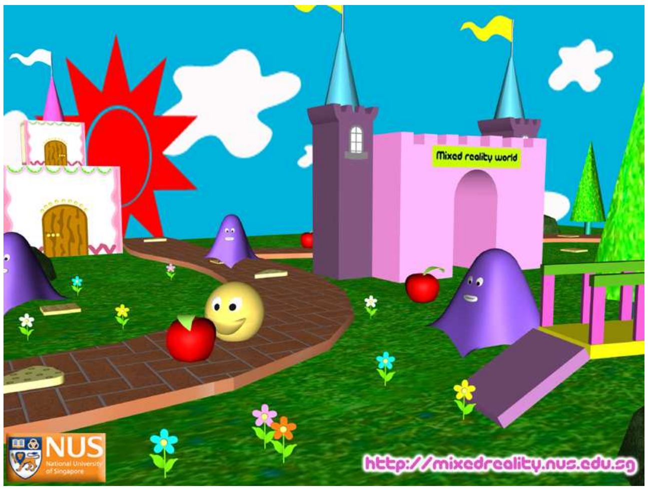
Mixed reality world
NUS National University of Singapore
http://mixedreality.nus.edu.sg

Figure 5.7: The colorful picture designed for the tracking of Pacman maze by using the natural features in the picture.

In this application, instead of searching for the initial matches all over the whole image, a smaller search window is defined based on the rotation information received from a tilt pad (Figure 5.8), which is attached at the back of the Pacman board (Figure 5.9). The tilt pad consists of two accelerometers and use a small battery as its power. It helps to define a small search window, which screens out some obvious erroneous matches in each incoming image. So the computational load when the system does the RANSAC looping will be minimized.

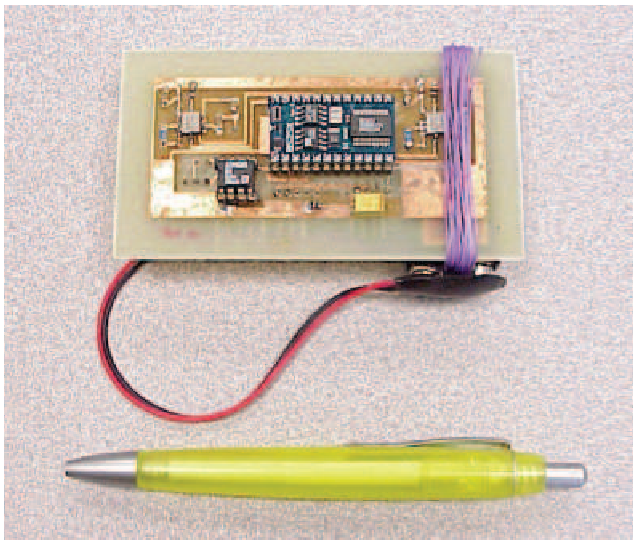

Figure 5.8: The tilt pad

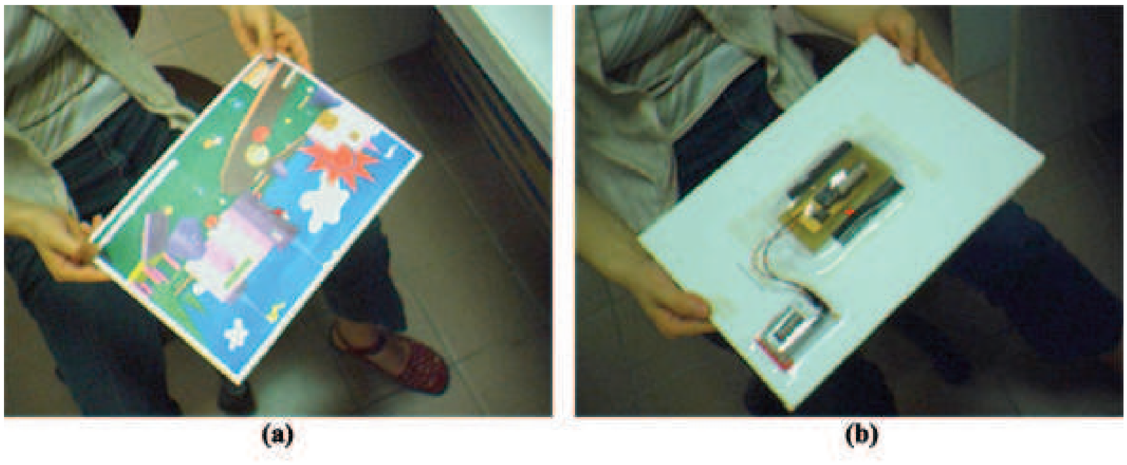

Figure 5.9: The board used in the Pacman game application. (a) The front view of the board: the colorful page to be tracked is pasted here. (b) The rear view of the board: the tilt pad is mounted here.

The reason we decided to incorporate these accelerometer's data to control the pacman movement in the maze, instead of only using the rotation information obtained from the natural feature tracking algorithm, is because the accelerometers always give very accurate angle values relative to the actual horizontal plane in the world coordinate. People are normally used to take this physical horizontal plane as the reference to decide what is so called "turning up", "turning down", "turning left" or "turning right". In contrast, the camera will only know the angle between the Pacman board to the camera 3D position in space by the vision-based algorithm. So it will become very difficult for the users to control the pacman in the maze without the tilt pad. However, the tilt pad may not be necessary for other applications, like the room decorative application and geographic labelling as we showed before. More details on the design and accuracy of the tilt pad can be found in [61].

As shown in Figure 5.10, when user tilts the board forward, the pacman (the yellow ball in the maze) will roll forward. When she tilts the board back, the pacman will also roll back. Same for the left and right directions. By tilting the board in different directions and controlling the tilting angle to the horizontal plane, user can determine the direction and speed of the pacman in the maze. So they can lead the pacman to eat all the berries, and run away from the ghosts. Since we are not using any special-purpose devices such as a mouse or keyboard, but just use the hands to hold Pacman maze and tile it to different angles to control the motion of the pacman, this application is also a very good example for Tangible User Interface as we mentioned in Chapter 3. The video clip of this game PacmanGame.avi can be found in the attached CD-ROM.

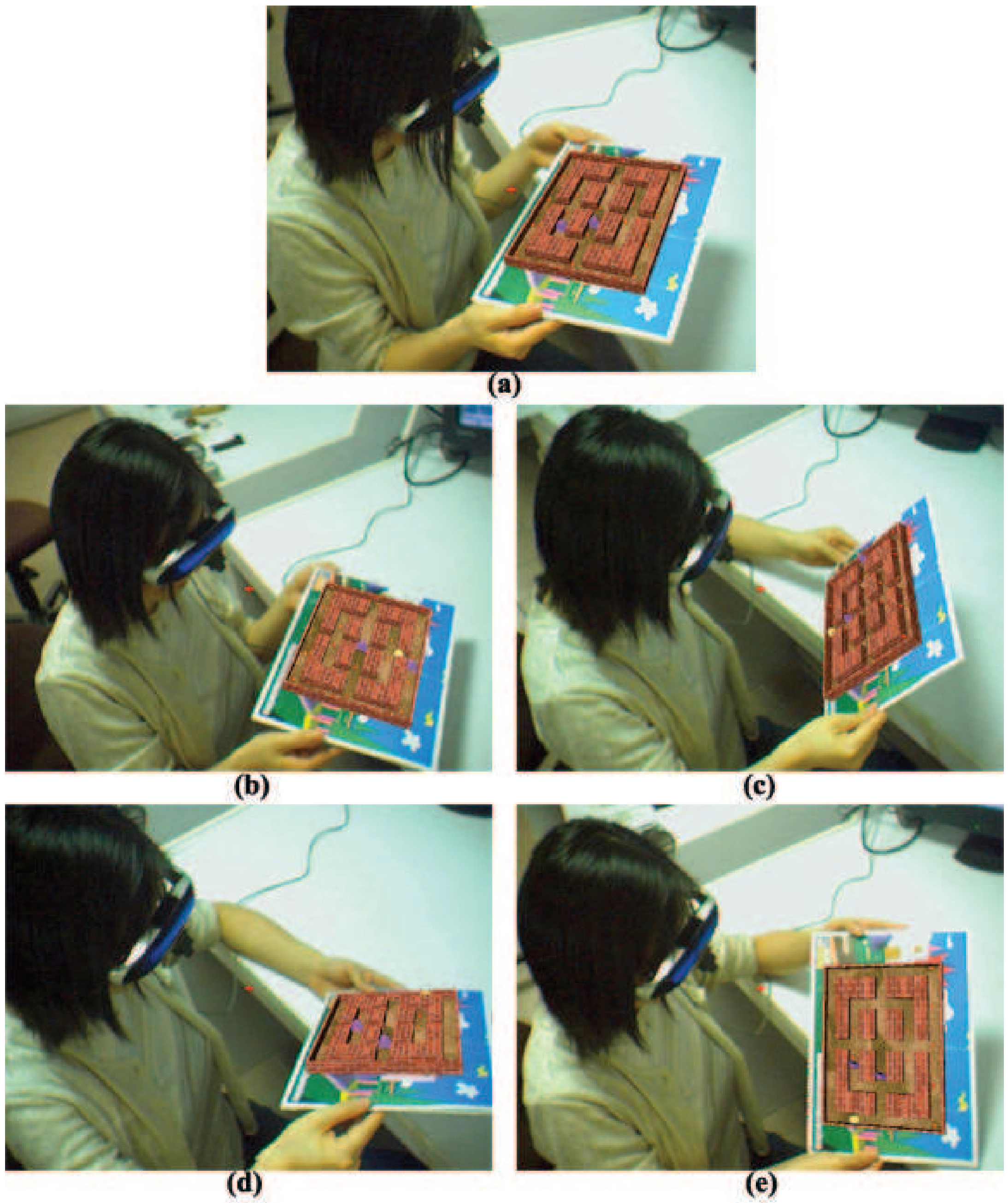

Figure 5.10: The user is playing the 3D Pacman game. (a) The user is holding the board, on which the colorful picture is pasted. The 3D Pacman maze is now augmented onto the board. (b) The user tilts the board forwards, the Pacman (presented by the yellow ball in the maze) goes forward, to eat the berries and avoid the ghosts. (c) The user tilts the board backwards. (d) The user tilts the board leftwards. (e) The user tilts the board rightwards.

Chapter 6

System Performance and Assessment

We restrict the remaining performance assessments to the real-time system based on calculating homographies. The fundamental-matrix based matching produces similar results over a wider range of image conditions, but with a considerably increased computational cost.

6.1 System Implementation Details and Performance

Typically 50-60 corners per image are identified, of which >80% are inliers to the final solution, depending on the amount of overlap of the two images. If less than 35% of corners are inliers we consider the calculation to have failed. We apply a number of heuristics to increase the system speed and accuracy: we repeat the RANSAC sampling up to 70 times, but immediately accept the solution if it has greater than 75% support and 15 iterations have already been computed. Using this criterion, the majority of frames are completed prematurely. In order to increase the average quality of the solutions, we ensure that the initial four points in the first image are spatially separated by at least 60 pixels. We also reject homographies that are near singular by testing the determinant. Singular matrices map the entire first image to a line or point in the second image.

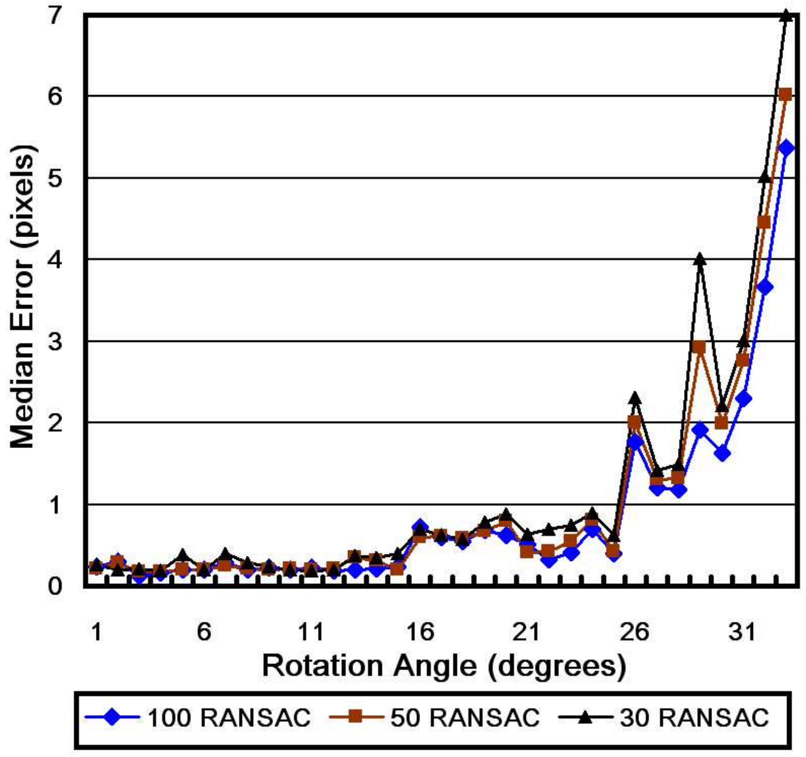

Figure 6.1: Placing of annotation is demonstrated to be accurate to below one pixel over 25 degree rotations for homography calculation. Performance degrades as the image overlap becomes negligible (camera field of view measured at 33 degrees).

For a static camera in an indoor environment, the homography was successfully calculated for 500/ 500 test frames in a 20 second sequence. For each frame, the center point of the image was transformed by the incoming homography. Since the camera is static, we expect it to remain in the same place. The mean deviation was <1 pixel per frame. Performance as a function of image rotation is assessed in Figures 6.1 and 6.2. In each case we matched 100 images from two static video streams of the same scene, where the camera had been rotated between capture. Two example frames are shown in the bottom panels of Figure 4.2. No prior information was given about the direction or magnitude of the camera movement. Performance is at or close to pixel accuracy across a wide range of distances. As shown in Figures 6.1 and 6.2, when the rotation angle is more than 31 degrees, the two adjacent image frames will only very little common contents, estimating a accurate homography will become very difficult. The median error can be more than 5 pixels, and percentage of successful trials will drop to almost zero.

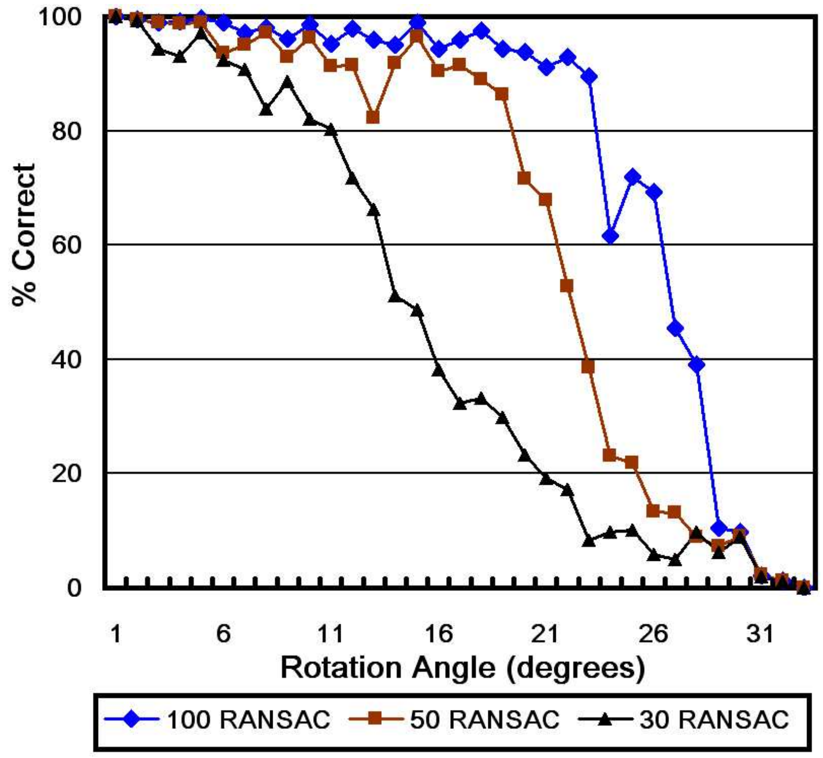

Figure 6.2: Percentage of successful trials for homography algorithm as a function of rotation angle. Performance extends to larger angles as the number of iterations of the robust estimation algorithm increases.

6.2 System Assessment

In this section we compare the three types of algorithm (based on computation of optical flow, fundamental matrix and homography) across a number of different criteria.

Generality: The fundamental matrix calculation is appropriate for rigid scenes under any camera translation and rotation and embodies the resulting constraints in the image flow. As previously discussed, the homography describes only a limited subset of image motions including those due to camera rotation, or any general camera movement in front of a planar surface. However, in practice the movement of image points due to small lateral translations, and moderate translations along the camera axis is well described by the homography.

We demonstrate this by taking a number of pictures of the same buildings from a variety of positions (see Figure 6.3). These buildings were far from planar and the camera position has moved 20 meters in each of 8 compass directions. By relaxing the threshold for inliers, we can calculate homographies reliably in almost all cases. Notably performance is better when movement is forward or backward rather than sideways. One must consider this test to be a worst case scenario - the system has no prior idea of the mapping. In our wearable application, we improve performance to near-perfect by calculating the homography to the previous frame (always possible) and using this to predict point position in the reference frame.

Since the motion flow pattern is not actually described by a homography, small but consistent biases appear in the feature position, and these generally become larger as we move away from the reference image. The maximum bias over this area was 6 pixels, which is clearly insignificant for a labeling application. The maximum standard deviation of the error around this bias position was of the order of 2 pixels as seen in the lower part of Figure 6.3.

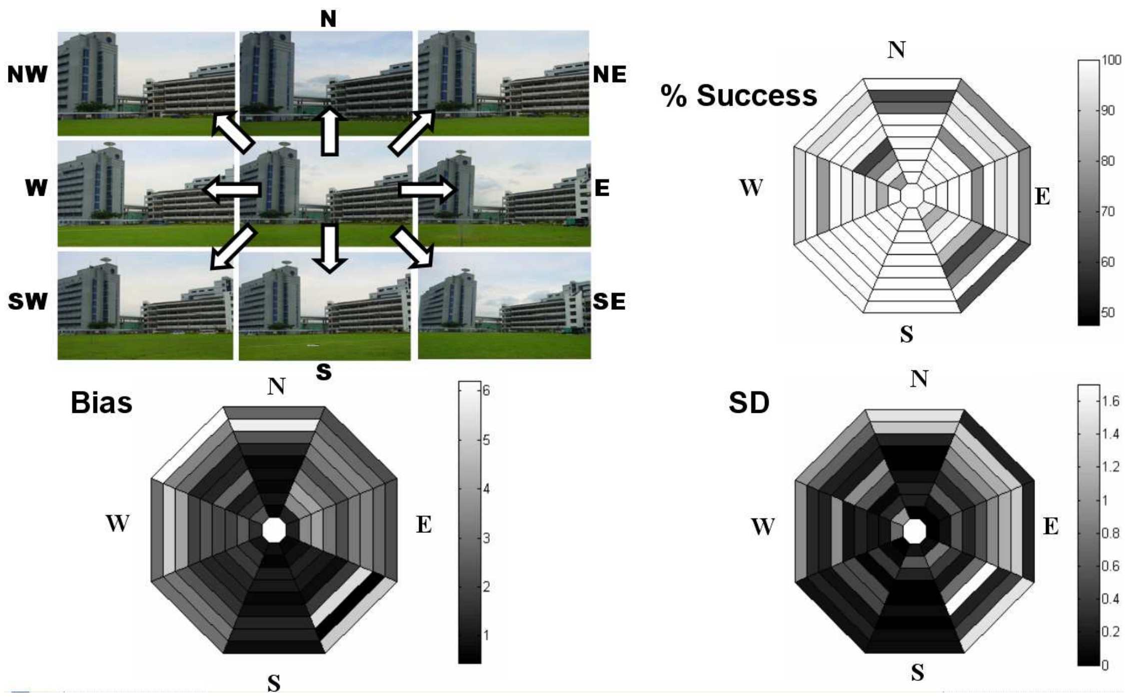

Figure 6.3: In order to test the robustness of the homography calculation where the conditions for this type of mapping are not met, we took 80 pictures of the same buildings at 2 meter intervals in 8 compass directions (top left). In each case we attempted to match the picture to a reference frame at the centre of this space 100 times, using 50 RANSAC repetitions. The proportion of successful trials is depicted in the top-right of the figure. We define a trial as successful if it mapped the top left corner of the right-most building to within 3 pixels of the median position across the 100 trials. Almost all trials were successful, even in these difficult conditions. Moreover, both the bias in position (induced because the situation is not actually described by a homography) and the jitter around this bias are small.

The optical flow algorithm does not place any constraints on image flow and hence is not optimal for geographical labeling which generally involves viewing a static world. However, it should be noted that algorithms based on optical flow can potentially cope with more general augmented reality applications where more than one independently moving object is considered.

Completeness: In the absence of *a priori* image depth information, the algorithm only predicts image velocity for corners matched during the initial calculation. A complete velocity map may be approximated by interpolation or calculated by a further dense matching process. The same is true for optical flow computations. The homography has the considerable advantage, that (when it can be computed), it predicts the exact corresponding point in the second image.

Robustness: The computation of the fundamental matrix and homography uses a robust estimation method [36]. Since both calculations involve finding constraints on the image flow, a simple test against these constraints establishes what proportion of the data agrees with the current estimate. This robustness to false matches allows the algorithms to triumph even when part of the image is obscured, has changed between frames, or is moving. It also means that the velocity field may be described by a homography when the scene is “mostly” planar. As long as the augmentations need to be superimposed upon the planar part of the scene, this model will suffice. In contrast, methods based on optical flow cannot be robust in the same way, since they do not impose any constraints on the image flow. Having said this, it is possible to take their output and use a robust algorithm to establish an ad hoc local flow model such as the region based affine flow model used in [37].

Speed: As previously described, the homography can be calculated reliably at camera frame rate. Since the bulk of the computation is due to the image processing (corner detection), the fundamental matrix calculation takes only slightly longer.

The principle reason for the decrease in speed is due to the fact that it requires 8 correct matches rather than 4 to calculate this entity. The random sampling algorithm takes longer on average to pick 8 correct matches in one iteration than it did to choose only 4. Moreover, we must now invert a system of eight equations for each candidate matrix. Optical flow methods are generally much slower as they require multiple calculations at different image scales.

Reliability: The homography and fundamental matrix provides a natural way to determine whether the image flow has been estimated correctly. The proportion of inliers to the model can be tested against a fixed threshold to determine the success of the procedure. There is no such way to determine how successful an optical flow calculation has been. The homography calculation may fail if the underlying assumptions (planarity or rotation) are not met. The fundamental matrix calculation may also fail if the image is not static. If the image does not contain sufficiently general structure (at least 3 planes), the fundamental matrix calculation may fail entirely as eight independent equations may not be found. However, for the purposes of the image motion field estimation, any of the matrices in the resulting null space will suffice equivalently.

The most obvious weakness of all of these systems is that they rely on the ability to find and track stable corners in the incoming video-stream. The system will not perform well if the camera view contains large blank walls or highly dynamic scenes such as crowds. These problems are ubiquitous in current vision-based tracking algorithms. The most obvious solution is to combine the system with data from other tracking modalities.

Range: The range of image displacements over which the algorithms succeed is an important factor. We assume that the systems have no a priori knowledge about the image motion. Optical flow calculation relies on local image gradients and hence naturally operates on a very small scale (only a few pixels). The range can be extended by using coarse-to-fine sampling schemes, but this increases the amount of computation. The performance of the homography-based matching procedure is illustrated as a function of rotation angles in Figure 6.1 and Figure 6.2. The fundamental matrix-based system gives similar results but requires more iterations. Performance can be improved by using information from the mapping in the previous frame to help search for matches.

Accuracy: An important criterion for assessment of these algorithms is the accuracy in the estimation of the velocity field. If we are calculating the velocity field with respect to a fixed reference frame each time, then accuracy of the order of a single pixel is desirable. If we are always matching to the previous frame, and attempting to maintain the position of the annotated information, then accuracy that is considerably sub-pixel is necessary, or else large errors will build up very quickly. We hence suggest that matching to reference images is desirable. This is easily achieved using the homography and fundamental matrix which can match over large ranges, but may be impractical with optical flow based calculations which cannot match over such large distances without further elaborations.

Chapter 7

Summary

Conclusion of this thesis work

For vision-based AR applications, accurate measurements of the camera pose relative to the real world are required so as to maintain the stable percept that the virtual object is part of the real scene. There are basically two approaches to this problem: by using the fiducial markers in the scene, and by tracking the natural features.

In the first part of this thesis work, a tangible user interface was developed based on the fiducial tracking approach, which is suitable for various desktop AR interactive applications. This new interface is universal. It has been merged with people's everyday interaction with the environment, such as picking and dropping. Thus, these interactions become intuitive and seamless because we would use the same tools to work with digital and real objects.

However, for most outdoor AR applications, the environment is normally unprepared, i.e. no fiducials are available in the scene. To solve this problem, a realtime AR registration system based on natural corner features in an arbitrary scene was developed in this thesis work. Two novel methods based on the fundamental matrix and the homography were presented. The experiment results show that they are superior to current methods across a wide range of criteria. These methods take two input frames, and optionally an a priori estimate of the velocity map and return the estimated motion field between the frames. We have demonstrated in that both two-dimensional and three-dimensional augmentation problems can be solved by these methods under certain constrains.

In this thesis work, we have also developed several realtime applications based on this proposed tracking algorithm, including geographical information augmentation, and indoor room decorative application. At the end, we assessed all three styles of the natural feature algorithms (optical flow, fundamental matrix and homography) across a number of criteria including robustness, speed and accuracy.

The videos from this thesis work, as well as some of the source code and tutorials for various applications mentioned in this paper can be found in the attached CD-ROM. Alternatively, all these information can also be downloaded from our lab website as listed in the Appendix attached.

Problem faced in this thesis work

Although the proposed vision-based tracking system is able to work in unprepared scenes, it must be noted that the viewed scene most be principally static and rigid. This is because camera pose information is extracted from the 2D motion of corner points i.e. corner matches and it is assumed that these changes in corner positions are due purely to camera movement and not due to movement of the corresponding 3D points. If this assumption is not satisfied, the recovered 2D motion field will not be an accurate representation of the actual camera motion. Also, the scene must be sufficiently textured so as to reliably identify corner points across images. Like any vision-based tracker, the proposed system will also fail if lighting conditions are unfavourable so that corner points cannot be reliably tracked across images.

Future work

One possible solution to the above problem of the proposed system is to use a hybrid approach to camera tracking which combines the use of 6DOF inertial sensors with the proposed vision-based method. These sensors can continue to track camera pose when the scene structure and lighting conditions are not favorable for the proposed system. The motion estimates from these sensors may be less accurate but can be refined by the vision-based tracker when scene conditions are once again favorable. Thus, one area of future research would be to develop the realtime framework for the hybrid tracking approach.

Appendix A

Contents of the CD-ROM

A CD-ROM is attached at the back of this thesis paper, which provides a soft copy version of this thesis, as well as the source code and result videos for the applications mentioned in the previous chapters. It has the following three main folders:

Folder One: Thesis

- A PDF version of the thesis — Thesis.pdf

Folder Two: Source Code and Tutorials

- Tangible AR Interaction — The Virtual Hello Kitty Garden
- Natural Feature Tracking AR Systems — Geographic Labelling/Overlaying Application
- Natural Feature Tracking AR Systems — Room Decorative Application
- Natural Feature Tracking AR Systems — Pacman Game Application
- Tutorials on these applications — Tutorial.txt

Folder Three: Videos

- Tangible AR Interaction — TangibleInteraction.avi
- 3D Augmentation on Planar Surface — Cube.avi
- Geographic Labelling Application — TextLabel.avi
- Geographic Overlaying Application — BuildingOverlay.avi
- Room Decorative Application — IndoorLight.avi
- Pacman Game Application — PacmanGame.avi

Appendix B

The Related Website Links

- The main site:
  http://mixedreality.nus.edu.sg/main(H).htm
- Tangible interaction project:
  http://mixedreality.nus.edu.sg/research-EMRI-infor.htm
- Tangible interaction project videos:
  http://mixedreality.nus.edu.sg/research-EMRI-videos.htm
- Natural feature tracking project:
  http://mixedreality.nus.edu.sg/research-NFT-infor.htm
- Natural feature tracking project videos:
  http://mixedreality.nus.edu.sg/research-NFT-videos.htm
- Various public demos and exhibitions of this Masters project work:
  http://mixedreality.nus.edu.sg/intro_photos.htm
- Source code and tutorials download site:
  http://mixedreality.nus.edu.sg/software.htm

Bibliography

[1] K. Xu, A. D. Cheok, K. W. Chia, and S. J. D. Prince, “Visual Registration for Geographical Labeling in Wearable Computing,” in *Proc. ISWC*, pp. 109–116, Seattle, USA, October 2002.

[2] S. J. D. Prince, K. Xu, and A. D. Cheok, “Robust Camera Tracking for Augmented Reality Based on Planar Homographies,” *IEEE Transactions on Computer Graphics and Applications*, vol. 22, pp. 39–45, November/December 2002.

[3] K. Xu, S. J. D. Prince, A. D. Cheok, Y. Qiu, and K. G. Kumar, “Visual Registration for Unprepared Augmented Reality Environments,” *Personal and Ubiquitous Computing*, vol. 7, pp. 287–298, 2003.

[4] P. Milgram and F. Kishino, “A Taxonomy of Mixed Reality Visual Displays,” *IEICE Transactions of Information Systems*, vol. E77-D, pp. 282–292, December 1994.

[5] R. T. Azuma, “A Survey of Augmented Reality,” *Presence*, vol. 6, no. 4, pp. 355–385, 1997.

[6] Brooks and P. J. Frederick, “The Computer Scientist as Toolsmith II,” *CACM*, vol. 39, no. 3, pp. 61–68, 1996.

[7] R. T. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent Advances in Augmented Reality," *IEEE Computer Graphics and Applications*, vol. 21, pp. 34–47, December 2001.

[8] M. Billinghurst, H. Kato, and I. Poupyrev, "The magic book: an interface that moves seamlessly between reality and virtuality," *IEEE Computer Graphics and Applications*, vol. 21, pp. 2–4, May/June 2001.

[9] W. Barfield and T. Caudell, *Fundamentals of wearable computers and augmented reality.* ISBN 0-805-82901-6, Lawrence Erlbaum Associates (LEA) Press, November 2000.

[10] W. Robinett, "Synthetic experience: A proposed taxonomy," *Presence: Teleoperators and Virtual Environments*, vol. 1, no. 2, pp. 229–247, 1992.

[11] M. Ward, R. Azuma, R. Bennett, S. Gottschalk, and H. Fuchs, "A demonstrated optical tracker with scalable work area for head-mounted display systems," in *Proceedings of 1992 Symposium on Interactive 3D Graphics*, pp. 43–52, Cambridge, MA, April 1992.

[12] H. Sowizral and J. Barnes, "Tracking position and orientation in a large volume," in *Proceedings of IEEE VRAIS '93*, pp. 132–139, Seattle, USA, September 1993.

[13] M. Bajura and U. Neumann, "Dynamic Registration Correction in Video-Based Augmented Reality Systems," *IEEE Computer Graphics and Applications*, pp. 52–60, September 1995.

[14] U. Neumann and Y. Cho, "A Self-Tracking Augmented Reality System," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pp. 109–115, Hong Kong, China, July 1996.

[15] A. State, G. Hirota, D. T. Chen, B. Garrett, and M. Livingston, "Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking," in *Proc. SIGGRAPH 96'*, pp. 429–438, New Orleans, USA, August 1996.

[16] R. A. Kancherla, P. R. Jannick, L. W. Donna, and G. Burdea, "A Novel Virtual Reality Tool for Teaching Dynamic 3D Anatomy," in *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95 (CVRMed '95)*, pp. 163–169, Nice, France, April 1995.

[17] I. N. Durlach and A. S. Mavor, *Virtual Reality: Scientific and Technological Challenges.* ISBN 0-309-05135-5, National Academy Press, 1995.

[18] M. Bajura, H. Fuchs, and R. Ohbuchi, "Merging Virtual Reality with the Real World: Seeing Ultrasound Imagery Within the Patient," *Computer Graphics*, vol. 38, July 1992.

[19] A. State, T. C. David, C. Tector, A. Brandt, H. Chen, R. Ohbuchi, M. Bajura, and H. Fuchs, "Case study: Observing a volume rendered fetus within a pregnant patient," in *Proceedings of IEEE Visualization '94*, pp. 364–368, Washington D.C., USA, October 1994.

[20] A. State, M. A. Livingston, G. Hirota, W. F. Garrett, M. C. Whitton, H. Fuchs, and E. D. Pisano, "Techniques for augmented-reality systems: Realizing ultrasound-guided needle biopsies," in *Proceedings of SIGGRAPH '96*, pp. 439–446, New Orleans, USA, August 1996.

[21] W. Grimson, T. Lozano-Perez, W. Wells, G. Ettinger, S. White, and R. Kikinis, "An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization," in *Proceedings of IEEE Confer-*

ence on Computer Vision and Pattern Recognition, pp. 430–436, Los Alamitos, USA, June 1994.

[22] W. Grimson, G. J. Ettinger, S. J. White, P. L. Gleason, T. Lozano-Perez, W. M. Wells, and R. Kikinis, "Evaluating and validating an automated registration system for enhanced reality visualization in surgery," in *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95*, pp. 3–12, Nice, France, April 1995.

[23] J. P. Mellor, "Realtime camera calibration for enhanced reality visualization," in *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95 (CVRMed '95)*, pp. 471–475, Nice, France, April 1995.

[24] W. Lorensen, H. Cline, C. Nafis, R. Kikinis, D. Altobelli, and L. Gleason, "Enhancing reality in the operating room," in *Proceedings of Visualization '93*, pp. 410–415, Los Alamitos, USA, October 1993.

[25] S. Feiner, B. MacIntyre, and D. Seligmann, "Knowledge-based Augmented Reality," *Communications of the ACM*, vol. 36, July 1993.

[26] T. P. Caudell and D. W. Mizell, "Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes," in *Proceedings of Hawaii International Conference on System Sciences*, pp. 659–669, Hawaii, USA, January 1992.

[27] D. Sims, "New Realities in Aircraft Design and Manufacture," *IEEE Computer Graphics and Applications*, vol. 14, March 1992.

[28] G. Fitzmaurice, "Situated information spaces: Spatially aware palmtop computers," *CACM*, vol. 36, July 1993.

[29] J. Rekimoto and K. Nagao, “The world through the computer: Computer augmented interaction with real world environments,” pp. 29–36, Pittsburgh, PA, November 1995.

[30] J. Rekimoto, “The magnifying glass approach to augmented reality systems,” in *Proceedings of ICAT ’95*, pp. 20–22, Makuhari Messe, Japan, November 1995.

[31] E. Rose, D. Breen, K. Ahlers, C. Crampton, M. Tuceryan, R. Whitaker, and D. Greer, “Annotating Real-World Objects Using Augmented Reality,” in *Proceedings of Computer Graphics International ’95*, pp. 357–370, Leeds, UK, June 1995.

[32] P. Maes, “Artificial Life Meets Entertainment: Lifelike Autonomous Agents,” *CACM*, vol. 38, November 1995.

[33] K. N. Kutulakos and J. Vallino, “Affine Object Representations for Calibration-Free Augmented Reality,” in *Proc. Virtual Reality Annual International Symposium (VRAIS)*, pp. 25–36, Santa Clara, USA, March 1996.

[34] Y. Cho and U. Neumann, “Multi-ring color fiducial systems and a detection method for scalable fiducial-tracking Augmented Reality,” in *Proc. First International Workshop on Augmented Reality*, p. 212, San Francisco, USA, November 1998.

[35] D. Dementhon and L. Davis, “Model based object pose in 25 lines of code,” *IJCV*, vol. 15, pp. 123–141, 1995.

[36] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Comm. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[37] U. Neumann and S. You, “Natural Feature Tracking for Augmented Reality,” *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 53–64, 1999.

[38] H. Kato and M. Billinghurst, “Marker Tracking and HMD Calibration for a Video Based Augmented Reality Conferencing System,” in *Proc. Int’l Workshop on Augmented Reality (IWAR 1999)*, pp. 85–94, Los Alamitos, Calif., October 1999.

[39] S. Guven and S. Feiner, “Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality,” in *Proc. of the Seventh International Symposium on Wearable Computers*, pp. 118–126, New York, USA, October 2003.

[40] W. Piekarski and B. H. Thomas, “An Object-Oriented Software Architecture for 3D Mixed Reality Applications,” in *Proc. of International Symposium on Mixed and Augmented Reality*, pp. 247–257, Tokyo, Japan, October 2003.

[41] M. Gorbet, M. Orth, and H. Ishii, “Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography,” in *Proceedings of Conference on Human Factors in Computing Systems (CHI ’98)*, pp. 49–56, Los Angeles, USA, 1998.

[42] A. Singer, D. Hindus, L. Stifelman, and S. White, “Tangible Progress: Less is More in Somewire Audio Spaces,” in *Proceedings of Conference on Human Factors in Computing Systems (CHI ’99)*, pp. 104–111, Pittsburgh, USA, 1999.

[43] R. Lindeman, J. Sibert, and J. Hahn, “Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments,” in *Proceedings of Conference on Human Factors in Computing Systems (CHI ’99)*, pp. 64–71, Pittsburgh, USA, 1999.

[44] H. Hoffman, "Physically Touching Virtual Objects Using Tactile Augmentation Enhances the Realism of Virtual Environments," in *Proceedings of Virtual Reality Annual International Symposium (VRAIS '98)*, pp. 59–63, Atlanta, USA, 1998.

[45] I. Poupyrev, N. Tomokazu, and S. Weghorst, "Virtual Notepad: Handwriting in Immersive VR," in *Proceedings of Virtual Reality Annual International Symposium (VRAIS '98)*, pp. 126–132, Atlanta, USA, 1998.

[46] H. Ihsii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms," in *Proceedings of Conference on Human Factors in Computing Systems (CHI '97)*, pp. 234–241, Los Angeles, USA, 1997.

[47] http://www.hitl.washington.edu/artoolkit/.

[48] K. Satoh, M. Anabuki, H. Yamamoto, and H. Tamura, "A Hybrid Registration Method for Outdoor Augmented Reality," in *Proc. 2nd IEEE and ACM Int'l Symp. Augmented Reality (ISAR'01)*, pp. 67–76, New York, USA, 2001.

[49] R. Azuma, J. W. Lee, Bolan Jiang, J. Park, S. You, and U. Neumann, "Tracking in Unprepared Environments for Augmented Reality Systems," *IEEE Computer and Graphics*, vol. 23, pp. 787–793, Dec. 1999.

[50] S. You, U. Neumann, and R. Azuma, "Hybrid Inertial and Vision Tracking for Augmented Reality Registration," in *Proceedings of Virtual Reality Annual International Symposium (VRAIS '99)*, pp. 260–267, Houston, USA, Mar. 1999.

[51] U. Neumann, S. You, Y. Cho, J. Lee, and J. Park, "Augmented reality tracking in natural environments," in *Proc. of the IEEE International Symposium on Mixed Realities*, San Francisco, USA, January 1999.

[52] D. Deriche and G. Giraudon, “A computational approach for corner and vertex detection,” *The International Journal of Computer Vision*, vol. 10, no. 2, pp. 101–124, 1993.

[53] http://sourceforge.net/projects/opencv/.

[54] C. J. Harris and M. Stephens, “A combined corner and edge detector,” in *Proc. 4th Alvey Vision Conferences*, pp. 147–151, Manchester, UK, 1988.

[55] R. Hartley, “In defense of the eight-point algorithm,” *IEEE PAMI*, vol. 19, pp. 580–593, October 1997.

[56] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge University Press, 2000.

[57] P. H. S. Torr and D. W. Murray, “The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix,” *Int. Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997.

[58] D. Eberly, *Rotation Representations and Performance Issues.* Magic Software, 6006 Meadow Run Court, Chapel Hill, NC 27516, 2002.

[59] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C.* Cambridge University Press, 2002.

[60] http://www.handykey.com/site/twiddler2.html.

[61] K. G. Kumar, A. D. Cheok, and S. Prince, “Micro-accelerometer based hardware interfaces for wearable computer mixed reality applications,” in *The 6th International Symposium on Wearable Computers-ISWC 2002*, pp. 86 –88, Seattle, USA, Oct. 2002.