Robust Registration of Virtual Objects for Real-Time Augmented Reality

by
Shahzad Malik

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

Master of Computer Science

The Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario, Canada

May 8, 2002

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

Robust Registration of Virtual Objects for Real-Time Augmented Reality

submitted by

Shahzad Malik

in partial fulfillment of
the requirements for the degree of
Master of Computer Science

_____

Dr. Frank Dehne
(Director, School of Computer Science)

_____

Dr. Gerhard Roth
(Thesis Supervisor)

_____

Dr. Prosenjit Bose
(Thesis Supervisor)

Abstract

Augmented reality is a technology which allows 2D and 3D computer graphics to be accurately aligned or registered with scenes of the real-world in real-time. The potential uses of this technology are numerous, from architecture and medicine, to manufacturing and entertainment. Vision-based techniques which augment objects onto predetermined planar patterns are considered the most promising approach for achieving accurate registrations, but the majority of the proposed methods fail to provide any robustness to significant changes in pattern scale, orientation, or partial pattern occlusion.

This thesis presents the design and implementation of a robust pattern-based augmentation system that addresses these problems, and analyzes its performance using standard consumer-level hardware. Known planar patterns are tracked in a real-time video feed, and virtual 2D and 3D objects are accurately augmented onto these patterns based on the plane's orientation. A method to achieve perspective-correct augmentations without the need for a manual camera calibration procedure is also described.

Acknowledgments

First, I would like to thank my thesis supervisors Gerhard Roth and Jit Bose. Without their support, this thesis would not have been possible.

It's not often that the path to a master's thesis is laid out on a yellow brick road, but Gerhard has provided just that. Without his expert guidance and knowledge, his continuous flow of ideas and improvements, his quick turnaround in providing feedback on my writing, and the non-stop stream of technical papers that he would leave for me on my desk, I would not have been able to come this far this quickly. I am especially grateful for the encouragement he has given me throughout this masters experience, as well as the insights he has given me regarding what it takes to become a successful researcher. Additionally, his discussions and opinions regarding government and politics provided a refreshing break from my C++ code.

I thank Jit for always being available to discuss thesis issues, course and degree requirements, and potential future PhD research topics and directions, even despite his busy schedule. I am most impressed by his humble demeanor, even though he is an expert in almost everything (including ping-pong and squash).

I've always had a bad habit of attempting to finish things as late as possible, and the submission and defense of this thesis was no exception. Both Linda Pfeiffer and Nicki Enouy deserve a huge thank you for scheduling my thesis examination at the last minute, and making sure all my degree requirements were met. Additionally, I would like to thank professors Eric Dubois, Wilf LaLonde, and Doron Nussbaum for agreeing to be on my defense committee with less than two weeks of preparation time.

The majority of my thesis research was carried out at the National Research Council (NRC), and I thank the members of the Computational Video Group for allowing me to make use of their facilities and equipment. Additionally, I'd like to thank Dmitry Gorodnichy, Chang Shu, and William Scott for reviewing my research paper submissions

and providing valuable feedback on my technical writing. Chris McDonald also deserves a thank you for providing a nice implementation of a blob-based augmented reality system that could be directly compared to my corner-based tracking system. Finally, Dmitry, Chang, and Philippe Massicotte (from the VIT group) deserve a thank you for providing some great competition during our after lunch ping-pong games in the Ballard Room.

I would also like to thank the Natural Sciences and Engineering Research Council (NSERC) for providing excellent funding to graduate students studying at Canadian universities. Without their financial support, my decision to attend graduate school would have been much more difficult.

Of course, without my parents, I would never have had the opportunity to explore a career in research. They have always taught me that a solid education and dedicated work ethic are the keys to success in life, and their endless love, support, and patience help me accomplish anything I set my mind to. They are the best parents I could have ever dreamed to have, and I dedicate this thesis to them.

Finally, I thank my wife, Jawairia, for being the most loving and wonderful person that I have ever known. She was willing to leave behind a loving family and life-long friends, and travel to the other side of the world just to be with someone who spends endless hours in front of the computer. I can't thank her enough for the sacrifices she has made, and I look forward to a lifetime together with her.

Contents

Abstract **iii**
Acknowledgements **iv**
Table of Contents **vi**
List of Tables **viii**
List of Equations **ix**
List of Figures **x**

1 Introduction **1**
1.1 Motivation 3
1.2 Augmentation Environment 6
1.2.1 Camera and Display Technology 6
1.2.2 Haptic User-Interface Device 9
1.3 Scope 10
1.3.1 Robustness Criteria 10
1.3.2 The Visual-Visual Registration Problem 11
1.4 Contributions 12
1.5 Thesis Overview 13

2 Related Work **14**
2.1 The Mathematics of Augmented Reality 14
2.1.1 Coordinate Systems 14
2.1.2 Camera Models 16
2.1.3 Camera Parameters 18
2.1.4 Camera Calibration 22
2.2 Registration Errors 24
2.2.1 Static Errors 24
2.2.2 Dynamic Errors 25
2.3 Sensor-based Registration Approaches 25
2.3.1 Magnetic Sensors 26
2.3.2 Inertial Sensors 26
2.4 Vision-based Registration Approaches 26
2.4.1 Tracking Known Patterns 27
2.4.2 Tracking Natural Features 29
2.5 Hybrid Registration Approaches 30
2.5.1 Magnetic and Vision 31
2.5.2 Inertial and Vision 33
2.6 Calibrated vs. Uncalibrated Registration 33
2.7 Open Problems 35

3 Planar Homographies for Registration **36**
3.1 Mathematical Background 36
3.2 Uncalibrated 2D Augmentations 38
3.3 Camera Parameter Estimation and 3D Augmentations 39

4 Robust 2D Pattern Tracker Design **43**
4.1 Defining a Pattern 43
4.2 Tracking System Overview 44
4.3 Search Mode 45
4.3.1 Candidate Region Detection 45
4.3.2 Convex Hull Fitting 48
4.3.3 Region Normalization 50
4.3.4 Pattern Identification 51
4.4 Tracking Mode 52
4.4.1 Coarse Corner Prediction 53
4.4.2 Subpixel Corner Detection 54
4.4.3 Homography Updating 56
4.5 Augmentation in 2D and 3D 57
4.5.1 Weighted Autocalibration 58
4.6 System Summary and Discussion 60

5 Analysis **62**
5.1 Performance 62
5.2 Self-Identification 63
5.3 Scale Invariance 64
5.4 Orientation Robustness 66
5.5 Occlusion Robustness 68
5.6 Lighting Robustness 70
5.7 Registration Stability 71
5.7.1 2D Registration 71
5.7.2 3D Registration 72

6 Conclusions **74**
6.1 Comparisons with Previous Work 74
6.2 Future Work 74
6.2.1 Occlusion Between Real and Virtual Objects 74
6.2.2 Virtual Lighting 75
6.2.3 Virtual Object Manipulation 76
6.2.4 Auditory and Haptic Devices 76
6.3 Summary 77

A Subpixel Corner Finding **78**

Bibliography **80**

List of Tables

Table 5.1 Distance (scale) tracking tolerance for test patterns 65

List of Equations

Equation 2.1	Perspective projection	17
Equation 2.2	Weak-perspective projection	18
Equation 2.3	Linking pixel coordinates and camera coordinates	19
Equation 2.4	Correcting for radial distortion	19
Equation 2.5	Transformation from world to camera coordinates	21
Equation 2.6	Relation between world coordinates and image coordinates	21
Equation 2.7	Intrinsic camera parameter matrix	21
Equation 2.8	Extrinsic camera parameter matrix	22
Equation 2.9	Projection equation in homogeneous matrix form	22
Equation 2.10	Perspective projection matrix	22
Equation 2.11	Weak-perspective camera matrix	22
Equation 3.1	2D-to-2D projective transformation	37
Equation 3.2	H as a simplified projection matrix	39
Equation 3.3	Orthogonality constraint #1	39
Equation 3.4	Orthogonality constraint #2	39
Equation 3.5	Orthogonality constraint #3	39
Equation 3.6	Equation 3.5 combined with Equation 3.2	40
Equation 3.7	Different representation of Equation 3.3	40
Equation 3.8	Different representation of Equation 3.4	40
Equation 3.9	Elimination of λ^2	40
Equation 3.10	Focal length f_u computation	40
Equation 3.11	Focal length f_v computation	40
Equation 3.12	Scale factor λ computation	40

List of Figures

Figure 1.1	Example of augmented reality for bridge building	2
Figure 1.2	Example of pattern-based augmented reality	5
Figure 1.3	Applications of augmented reality	5
Figure 1.4	Monitor-based display technology	7
Figure 1.5	Video see-through display technology	8
Figure 1.6	Optical see-through display technology	9
Figure 2.1	Coordinate systems in augmented reality	15
Figure 2.2	Perspective camera model	17
Figure 2.3	Weak-perspective camera model	18
Figure 2.4	Transformation between camera and world coordinates	20
Figure 2.5	Camera calibration pattern	23
Figure 2.6	ARToolkit augmentation	27
Figure 2.7	Restoration of a matrix pattern from a video frame	28
Figure 3.1	World coordinate system induced by a planar pattern	36
Figure 3.2	Uncalibrated registration of a 2D video image	39
Figure 3.3	Automatically calibrated 3D augmentation examples	41
Figure 3.4	3D augmentation of two checkered cubes	42
Figure 4.1	Example of a 64x64 black and white pattern with corner features	44
Figure 4.2	Basic flow of the tracking system	45
Figure 4.3	Binary thresholding example	46
Figure 4.4	Dynamic binary image quantization using histograms	47
Figure 4.5	Black image features in a binary image	49
Figure 4.6	Region orientation and vertex ordering of the convex hull	51
Figure 4.7	Corner prediction using a homography	53
Figure 4.8	Multiple potential corners in a single search window	54
Figure 4.9	Augmentation system outline	58
Figure 5.1	Corner Tracking Performance	63
Figure 5.2	Constrained pattern arrangement allowing up to 2^{12} unique patterns	64
Figure 5.3	Test patterns for tracking experiments	65
Figure 5.4	Allowable range of distances for a 20-corner pattern	66
Figure 5.5	Tracking under severe rotation	67
Figure 5.6	Tracker rotation tolerance	68
Figure 5.7	Example of pattern occlusion	68
Figure 5.8	Corner recovery from occlusion after pattern motion	70
Figure 5.9	Continuous corner tracking under lighting changes	71
Figure 5.10	Variance between computed and weighted focal lengths	73
Figure A.1	Four-connected neighbourhood for a corner pixel s	79

Chapter 1

Introduction

The sights and sound of a cool fall afternoon bring a sense of calm to the mind and soul. Red and yellow leaves flutter in the gentle breeze, a frisky little squirrel collects acorns for the pending winter, a flock of geese fly south in unison overhead, and a 20-foot T-Rex gently sips water from the clear flowing creek.

Hollywood movies have been merging computer-generated imagery with scenes of the real world for several years now, and the results are so realistic that it is sometimes difficult to differentiate between the real and the virtual. From the synchronized interactions between man and dinosaur in Jurassic Park, to the re-creation of the *Titanic* in the movie of the same name, special effects artists seem to have mastered the art of seamlessly combining photo-realistic virtual 3D objects with pre-recorded 2D video footage.

While the end results are fantastic, the process by which movie artists merge a 3D object into the real world is extremely time-consuming. Even for a small ten-second clip, a dozen or so computer artists and animators can spend hours per frame in their attempts to perfectly augment the virtual objects into the video sequence. The obvious question is whether this merging of virtual 3D objects with the real world can be done at interactive rates such that we can view and manipulate the 3D imagery in real-time. Augmented reality is one such approach to achieving this goal.

Unlike virtual reality (VR), which encompasses a user in a completely computer-generated environment, augmented reality (AR) is a technology that attempts to enhance a user's view of the real environment by adding virtual objects, such as text, 2D images, or 3D models, to the display in a realistic manner. The motivation behind AR is that a user's existing visual and spatial skills can be leveraged in order to interact with computer-generated objects, or to receive additional information about real-world

objects. For example, Figure 1.1a shows an engineer's original view of a city from the other side of a river, while Figure 1.1b shows the same view enhanced using augmented reality technology. The result is that the viewer is now able to preview a new bridge design, as it would appear in its proposed natural environment, without actually physically erecting the bridge.

While there are many aspects to consider when creating an augmented reality application, one of the most difficult is precisely calculating the user's viewpoint in real-time so that the virtual objects are exactly aligned with the real-world objects [KATO00a]. For example, to automatically modify the scene in Figure 1.1a, the computer system must have some knowledge of the location and orientation of the surrounding environment, from the user's perspective, before it can properly augment the new virtual bridge over top of the real scene. Additionally, if the user wishes to see the new bridge from different angles and viewpoints so that the bridge appears to be a natural part of the environment, the system must be able to continually perform the augmentation in real-time. This precise alignment and synchronization of virtual objects with the real world is referred to as the *registration* process.

(a) (b)

Figure 1.1 – Example of augmented reality for bridge building.
(a) The untouched scene showing a view of a proposed bridge location.
(b) The scene has been enhanced with an augmented reality model of the proposed bridge, allowing an engineer or city planner to preview the bridge before physically erecting it.

This thesis concerns itself with finding robust solutions to these key augmented reality issues, specifically the precise registration of virtual objects over a real-time video feed. The next section describes additional motivation for this work, while Section 1.2 provides an overview of the various components in typical augmented reality environments. Section 1.3 then describes the scope of the research, and Section 1.4 notes the key contributions that will be made from our research. Finally, Section 1.5 outlines the rest of this thesis.

1. 1 Motivation

As computers become increasingly smaller and more powerful, and as demand for interesting consumer electronics devices continues to increase, augmented reality has the potential to become the "killer application" that computer vision researchers have been waiting for. Consider visiting a foreign city for the very first time, and not having any idea of where you are, or where you need to go. Instead of consulting your dictionary on how to ask for directions in the local language, you instead put on your pair of sunglasses and immediately your surroundings are no longer so foreign. With the built-in augmented reality system, your sunglasses have converted all of the real-world signs and banners into English. As you move or turn your head, the translated signs all maintain their correct position and orientation, and additional directional arrows and textual cues guide you towards your desired destination.

Or consider a medical student training to become a heart surgeon. Instead of simply learning from textbooks and training videos, the student can apply his or her knowledge in an augmented reality surgery simulation. The entire operation can thus be simulated from start to finish in a realistic emergency room setting using computer-generated images of a patient, as well as force-feedback medical tools and devices to provide a true-to-life experience.

While these seem like scenarios from a science fiction movie, they aren't necessarily that far-fetched. The key to creating an effective augmented reality experience is mimicking

the real world as closely as possible. In other words, from a user interface perspective, the user should not have to learn to use the augmented reality system but instead should be able to make use of it immediately using his or her past experiences from the real world. Clearly, the visual aspect of augmented reality is a critical component in depicting this seamless environment, and the registration process thus plays a central role.

Consider the augmented reality surgery training system described earlier. If the registration process were inaccurate, the student would develop skills that would be effective in the augmented reality environment, but not necessarily in a real world operating room. Similarly, for the sign translation sunglasses, an inaccurate registration could cause the augmented signs or banners to oscillate or drift from their intended positions, resulting in severe motion sickness for certain users.

Computer vision researchers have been experimenting with real-time vision-based solutions to the registration problem for a few years. The most promising of these involve tracking known patterns of features on planar surfaces from frame to frame, and using the extracted positions of these features to perform the augmentation of virtual objects onto the plane [KLIN99]. Figure 1.2 shows an example of pattern-based augmentation. By tracking these pre-determined patterns, researchers hope to gain insights into the ultimate goal of tracking natural features in arbitrary environments.

To further complicate the situation, the registration process must be fast enough for the augmentation of the virtual objects to be done in real-time on low-cost hardware. Clearly, in order for augmented reality applications to become widely accepted by the mass consumer market, we must first find efficient and reliable solutions to the real-time registration problem. Only then can the potentials of augmented reality in fields such as medicine, construction, manufacturing, and entertainment [OHSH98, AZUM01, TAMU01] be fully realized (see Figure 1.3).

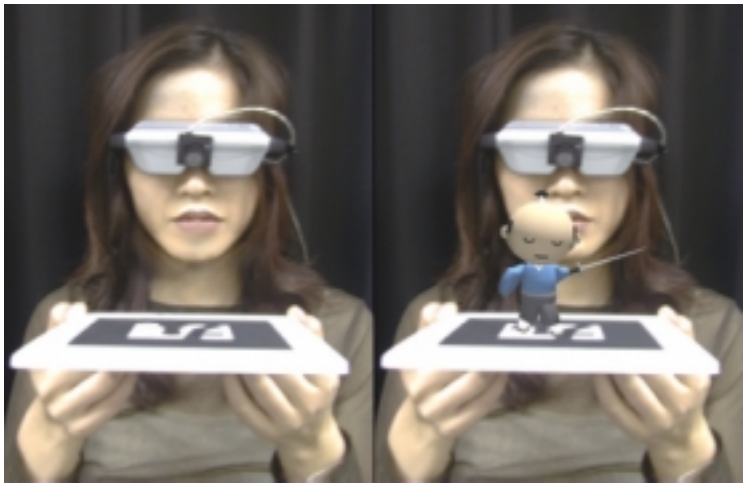


Figure 1.2 – Example of pattern-based augmented reality [KATO00a]. (a) The original scene, showing a piece of paper containing a predefined pattern. (b) The same scene after augmenting a virtual object on top of the known pattern.

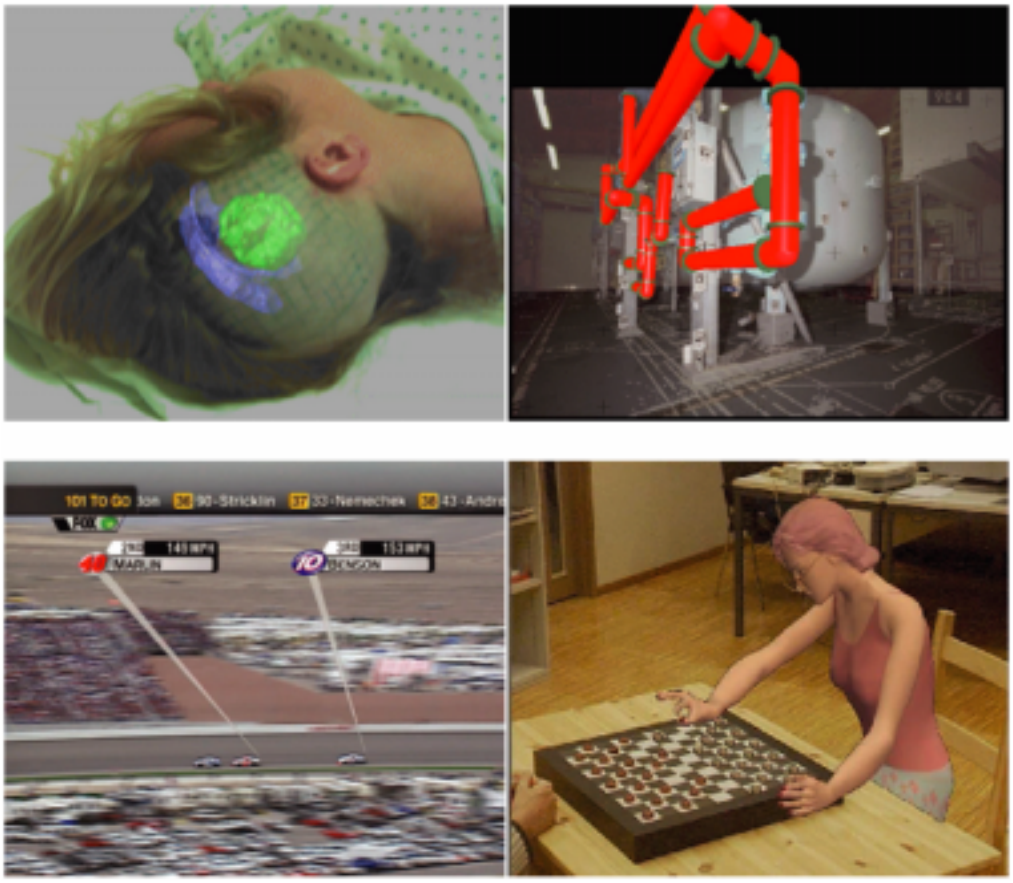


Figure 1.3 – Applications of augmented reality. Clockwise from top left: A surgeon's "X-Ray Vision" of a patient's brain; a virtual pipe superimposed onto a real industrial pipeline based on 2D floor plans [AZUM01]; autonomous virtual human playing checkers in real-time [BROL01]; annotations for the race-cars in a live sports broadcast [AZUM01].

1.2 Augmentation Environment

A typical augmented reality environment consists of a camera device, a display device, and in some cases a user-interface device to interact with the virtual objects.

1.2.1 Camera and Display Technology

In order to combine the real world with virtual objects in real-time we must configure camera and display hardware. The three most popular display configurations currently in use for augmented reality are Monitor-based, Video See-through and Optical See-through [AZUM97a, VALL98].

Monitor-based Display

The simplest approach is a monitor-based display, as depicted in Figure 1.4. The video camera continuously captures individual frames of the real world and feeds each one into the augmentation system. Virtual objects are then merged into the frame, and this final merged image is what users ultimately see on a standard desktop monitor. The advantage of this display technology is its simplicity and affordability, since a consumer-level PC and USB or FireWire video camera is all that is required. Additionally, by processing each frame individually, the augmentation system can use vision-based approaches to extract pose (position and orientation) information about the user for registration purposes (by tracking features or patterns, for example). However this simplicity comes at the expense of immersion. Clearly, viewing the real world through a small desktop monitor limits the realism and mobility of the augmented world. Additionally, since each frame from the camera must be processed by the augmentation system, there is a potential delay from the time the image is captured to when the user actually sees the final augmented image. Finally, the quality of the imagery is limited by the resolution of the camera.

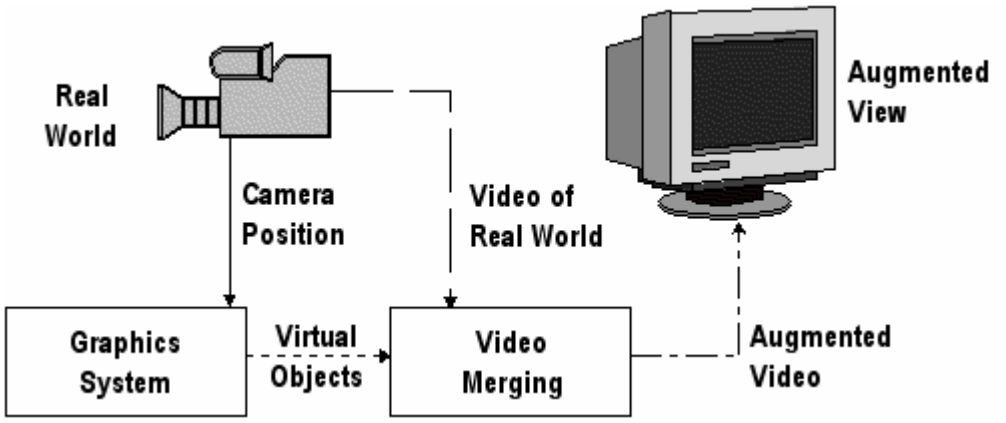


Figure 1.4 – Monitor-based display technology [VALL98]

Video See-through Display

In order to increase the sense of immersion in virtual reality systems, head-mounted displays (HMD) that fully encompass the user's view are commonly employed. There are two popular methods to bring HMDs into the augmented reality environment. Figure 1.5 shows a schematic for a video see-through augmented reality system. In this configuration, the user does not see the real world directly, but instead only sees what the computer system displays on the tiny monitors inside the HMD. The difference between this and a virtual reality HMD is the addition of video cameras to capture images of the real world. While this configuration is almost identical to the monitor-based technology in terms of functionality, the use of a stereo camera pair (two cameras) allows the HMD to provide a different image to each eye, thereby increasing the realism and immersion that the augmented world can provide. Like the monitor-based setup, the video see-through display is prone to visual lags due to the capture, processing, augmentation, and rendering of each video frame. Additionally, a large offset between the cameras and the user's eyes can further reduce the sense of immersion, since everything in the captured scenes will be shifted higher or lower than where they should actually be (with respect to the user's actual eye level).

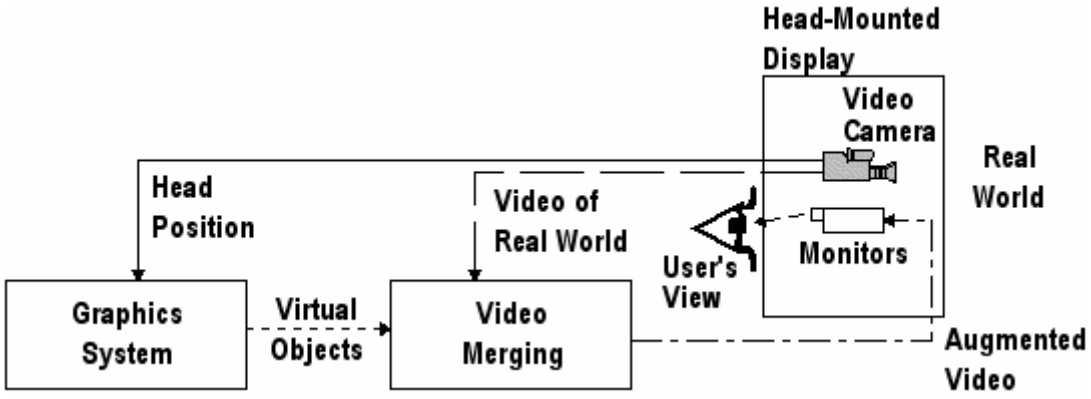


Figure 1.5 – Video see-through display technology [VALL98]

Optical See-through Display

The other popular HMD configuration for augmented reality is the optical see-through display system, as depicted in Figure 1.6. In this setup, the user is able to view the real world through a semi-transparent display, while virtual objects are merged into the scene optically in front of the user's eyes based on the user's current position. Thus when users move their heads, the virtual objects maintain their positions in the world as if they were actually part of the real environment. Unlike the video see-through displays, these HMDs do not exhibit the limited resolutions and delays when depicting the real world. However, the quality of the virtual objects will still be limited by the processing speed and graphical capabilities of the augmentation system. Therefore, creating convincing augmentations becomes somewhat difficult since the real world will appear naturally while the virtual objects will appear pixilated. The other major disadvantage with optical see-through displays is their lack of single frame captures of the real world, since no camera is present in the default hardware setup. Thus position sensors within the HMD are the only facility through which pose information can be extracted for registration purposes[1]. Some researchers have proposed hybrid solutions that combine position sensors with video cameras in order to improve the pose estimation; Section 2.4 discusses these hybrid solutions in more detail.

[1] Chapter 2 describes position sensors for registration purposes in more detail.

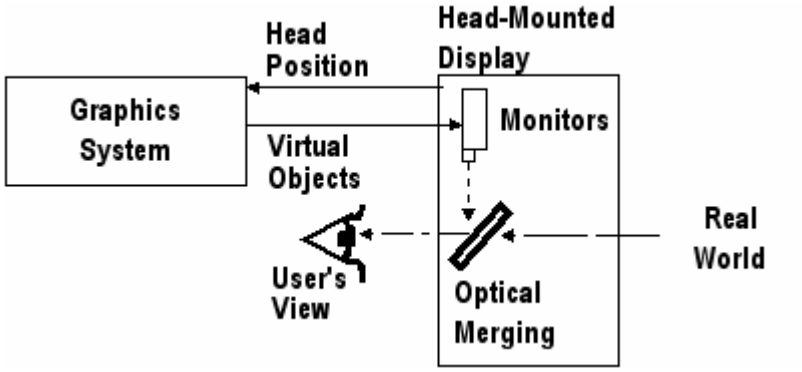


Figure 1.6 – Optical see-through display technology [VALL98]

1.2.2 Haptic User-Interface Device

Similar to virtual reality systems, augmented reality requires some facility through which a user can physically interact with the virtual objects. Haptic input devices provide this facility, but with the added ability to provide touch sensations to the user.

Consider a virtual coffee mug that has been augmented onto a user's real desk. With a specially designed glove, a user could physically grab hold of the coffee mug and hold it in his or her hand. With sensors in the palm and fingertip area of the glove, the user could feel the weight of the virtual mug as well as sense the texture across the mug's surface. Further, the augmented environment could simulate gravity as well, and thus releasing the virtual mug would send it crashing to the ground. Therefore, if simulated correctly, the user would not be able to distinguish this virtual mug from a real one.

While conceptually very simple, these haptic devices are under intensive research since recreating realistic physical sensations is a difficult problem. Many different devices have been proposed, from the above mentioned force-feedback gloves to full-size bodysuits that apply forces to a user's arms and legs. A detailed survey of the various haptic user interfaces that are applicable to augmented reality environments is provided in [SRIN97].

1.3 Scope

The focus of this thesis is on the registration process of augmented reality, specifically as it relates to the augmentation of 2D and 3D objects onto planar surfaces containing patterns of features. Our hypothesis is that the level of reliability or robustness that can be achieved will largely depend upon the features we choose to track, as well as the arrangement of these features in our planar patterns. Therefore, our research will attempt to analyze what constitutes a reliable pattern that is stable enough for high-quality, real-time augmentation.

1.3.1 Robustness Criteria

The robustness of a set of features will be assessed based on the following criteria:

Speed

For the purposes of real-time augmentation, the speed of our registration process is critical. Even if we can fulfill any of the other robustness criteria, our feature tracking system must still be able to run at interactive rates on consumer-level hardware.

Self Identification

If we are to augment virtual objects onto prearranged planar patterns, then we must be able to uniquely identify the pattern layout such that the proper association with a virtual object can be made.

Scale and Orientation Invariance

If a pattern layout is only reliable or identifiable at close range, then the augmentation of a virtual object onto that pattern will have a tendency to jitter severely or eventually disappear as a user gradually moves away from the object. Conversely, a pattern layout that is reliable only from far distances will cause an augmented object to jitter or disappear when a user attempts to examine it closely. Ideally, we would like a pattern arrangement that can robustly register the object with the environment both up close as

well as from a reasonably far distance. Similarly, a pattern that is not reliable or identifiable when viewed at extreme orientations (i.e. not completely front-facing) limits the ability to view augmented objects.

Robust to Occlusion

Clearly, for pattern-based augmented reality, attempting to examine an augmented object close up may cause some patterns to be out of the user's field of view. Similarly, if the user simply waves his or her hand over top of the pattern, some features may become temporarily occluded. In a perfect world, we would like to have a pattern layout that can compensate for these minor occlusions such that the virtual object continues to be augmented onto the display, at the right position with little or no jitter.

Robust to Lighting Changes

Similar to the occlusion problem, major changes in lighting could affect an augmentation system's ability to detect certain features in a pattern layout. Thus a pattern that can handle dynamic lighting changes is a major consideration when it comes to robustness in augmented reality.

1.3.2 The Visual-Visual Registration Problem

Registration is defined as the *precise alignment and synchronization of two or more sensory elements* [AZUM97b]. Thus far we have presented an introduction to the registration problem as it relates to accurately aligning virtual objects with the real world in augmented reality displays. However, this only addresses the visual-visual registration problem. In augmented reality, accurate registration between other sensory elements, such as haptic and auditory (and in the future possibly even taste and olfactory), is also crucial. Thus if a user swings a real baseball bat at a virtual baseball, an appropriate sound and force should be presented at the exact moment contact is made between the bat and the ball.

The quality of a user's experience with a virtual or augmented world is directly related to the quality of the registration process. Even if there is a slight delay between any of the sensory elements, the user's immersion in the world is negatively impacted. Since system delays are nearly impossible to avoid, the trick is to reduce these delays to an acceptable level so that they are not noticeable and do not reduce the quality of immersion. In augmented reality, the visual-visual registration errors are much more critical than other registration errors (such as the visual-auditory or visual-haptic errors) [AZUM97b]. Thus our research will focus on the visual-visual registration problem since it is the most critical for creating an effective augmented environment[2].

1.4 Contributions

This thesis presents a solution for real-time augmented reality that can operate on low-cost consumer-level hardware. The key contributions [MALI02a, MALI02b] are:

- A system architecture and implementation overview for a real-time augmented reality system that can augment virtual objects onto known patterns of planar features at real-time frame rates.

- An analysis of the proposed augmentation system, outlining what features allow for stable registrations, and what pattern layouts are self-identifying, provide invariance to scale, and are robust to occlusion and lighting changes.

- An automatic and stable camera calibration method that is specially suited to our pattern-based augmented reality system[3].

[2] For the remainder of this thesis, we will refer to the *visual-visual registration problem* as simply the *registration problem*.
[3] Chapter 2 discusses camera calibration in more detail.

1.5 Thesis Overview

The rest of this thesis is organized as follows:

Chapter 2 reviews previous work that our research draws upon, and also presents some of the mathematical principles underlying augmented reality technology.

Chapter 3 discusses planar homographies, which are extremely useful for augmentations of virtual 2D and 3D objects onto planar patterns.

Chapter 4 details the pattern-based augmented reality system that we have developed, focusing on the advantages, disadvantages, and tradeoffs of the 2D feature-tracking component. An overview of the system as it processes a real-time video feed into a fully augmented 2D or 3D scene is described.

Chapter 5 analyzes the implementation of the real-time augmented reality system in terms of robustness, registration stability and performance.

Chapter 6 concludes the thesis by summarizing the contributions made, comparing our results to previous work, and proposing areas for future research.

Chapter 2

Related Work

In this chapter, we review prior work in the field of augmented reality from which our research draws upon. We start by providing an introduction to the mathematical concepts of augmented reality, and describe how they relate to the registration problem. We then present a brief survey of various proposed solutions to the registration problem in the current augmented reality literature. We conclude by briefly describing some open issues.

2.1 The Mathematics of Augmented Reality

Before we can discuss the various solutions that have been proposed to solve the registration problem, we need to review some key mathematical ideas.

2.1.1 Coordinate Systems

The mathematical nature of the registration problem that has to be solved is depicted in Figure 2.1. The following three transformations, as described in [VALL98], that all augmented reality applications need to consider are Object-to-world, World-to-camera, and Camera-to-image plane.

Object-to-world ($\mathbf{M_O}$)

Assuming that we have a virtual object centered on its own local coordinate system, $\mathbf{M_O}$ will specify the transformation from this local system into a position and orientation within the world coordinate system that defines the real scene.

World-to-camera (M_C)

The $\mathbf{M_C}$ transformation specifies the position and orientation (pose) of the video camera that is being used to view the real scene, allowing points in the real world to be specified in terms of the camera's origin.

Camera-to-image plane (M_P)

The $\mathbf{M_P}$ transformation defines a projection from 3D to 2D such that camera coordinates can be converted into image coordinates for final display onto a monitor or HMD.

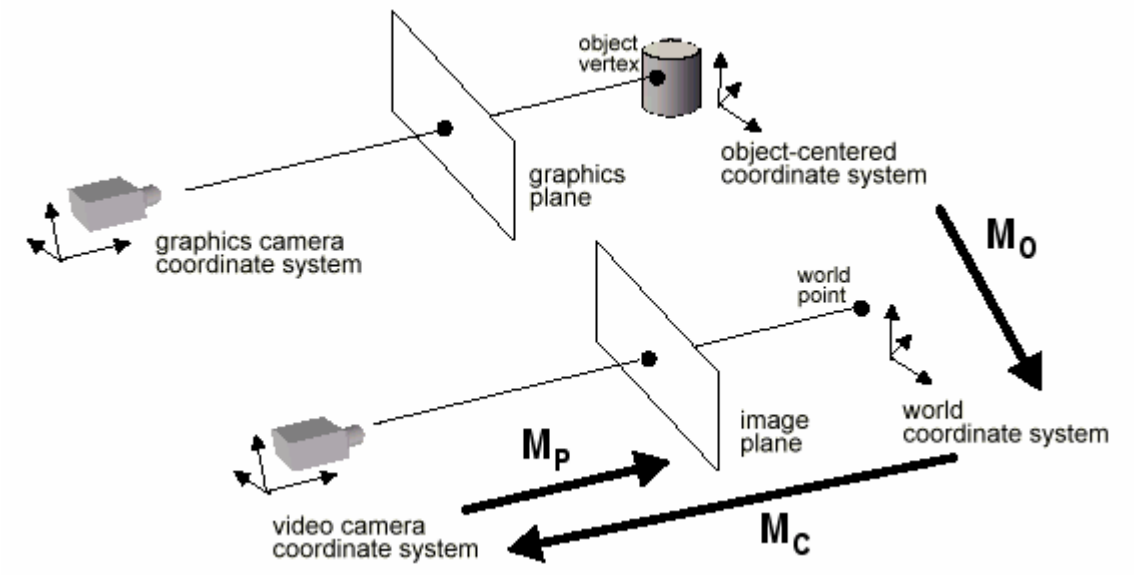


Figure 2.1 – Coordinate systems in augmented reality [VALL98].

In order for an augmented reality application to correctly render a virtual 3D object over top of a real scene, the above three geometric transformations have to be accurate. An error in any one of the relationships will cause the registration to be inaccurate, reducing the realism of the final augmented scene (see Section 2.2 for more information on these registration errors).

Since the virtual 3D objects will be rendered using standard 3D graphics hardware, it follows that they be represented using traditional computer graphics data structures. The surface of our virtual object can thus be represented as a triangular mesh, which consists

of a set of 3D vertices and a set of non-overlapping triangles connecting these vertices [FOLE90].

Using homogeneous coordinates, the obvious approach to augmenting these virtual objects requires that we determine the 2D projection [*u, v, h*] of a 3D point in Euclidean space [*x, y, z, w*] using the following equation:

$$[u\ v\ h]^T = \mathbf{M_{P(3x4)}}\ \mathbf{M_{C(4x4)}}\ \mathbf{M_{O(4x4)}}\ [x\ y\ z\ w]^T$$

The following sections will discuss ideas from projective vision that allow us to explicitly determine the $\mathbf{M_P}$, $\mathbf{M_C}$, and $\mathbf{M_O}$ transformations.

2.1.2 Camera Models

Assuming we have a [*x, y, z*] vertex in camera coordinates, projective geometry allows us to define the transformation $\mathbf{M_P}$ which can convert this 3D point into 2D image space.

The Perspective Camera

Figure 2.2 shows the *perspective* or *pinhole* camera model, which is considered the most common geometric model for video cameras [TRUC98]. The *optical axis* is defined as the line through the *center of focus* (a 3D point), which is perpendicular to the *image plane*. The distance between the *image plane* and the *center of focus* is referred to as the *focal length (f)*. The *principal point* is the intersection of the *optical axis* and the *image plane*. Assuming we have any other point **P** = [*X, Y, Z*] in 3D, and if we consider the *image plane* to define our 2D image, then the 2D projection of **P** is the intersection between the *image plane* and the line through the *center of focus* and **P**, denoted by **p** = [*x, y, z*]. In other words, we have

$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z}$$

Equation 2.1 – Perspective projection

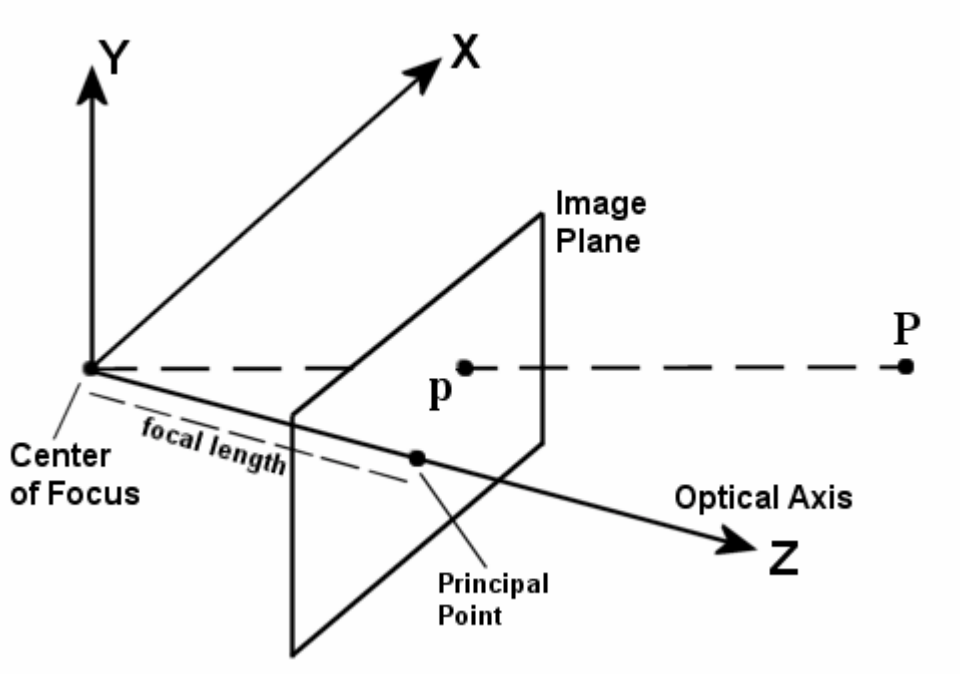


Figure 2.2 – The Perspective Camera Model

The Weak-Perspective Camera

Since the perspective projection is a non-linear mapping, it tends to make vision problems difficult to solve [XU96]. A commonly used approximation to the perspective camera model that simplifies certain computations is the *weak-perspective* camera. If, for any two points in a scene, the relative distance along the optical axis, δz, is significantly smaller than the average depth, Z_{avg}, of the scene, then the approximation holds [TRUC98]. Typically, $\delta z < (Z_{avg} / 20)$. Conceptually, we can think of the projection as a two-step projection [XU96]. The first is a projection of the object points onto a plane which goes through Z_{avg}. The second is a uniform scaling of the Z_{avg} plane onto the image plane (see Figure 2.3). Mathematically, we have

$$x = f\frac{X}{Z_{avg}}$$
$$y = f\frac{Y}{Z_{avg}}$$

Equation 2.2 – Weak-perspective projection

Typically, Z_{avg} can be the centroid of some small object in a scene [XU96].

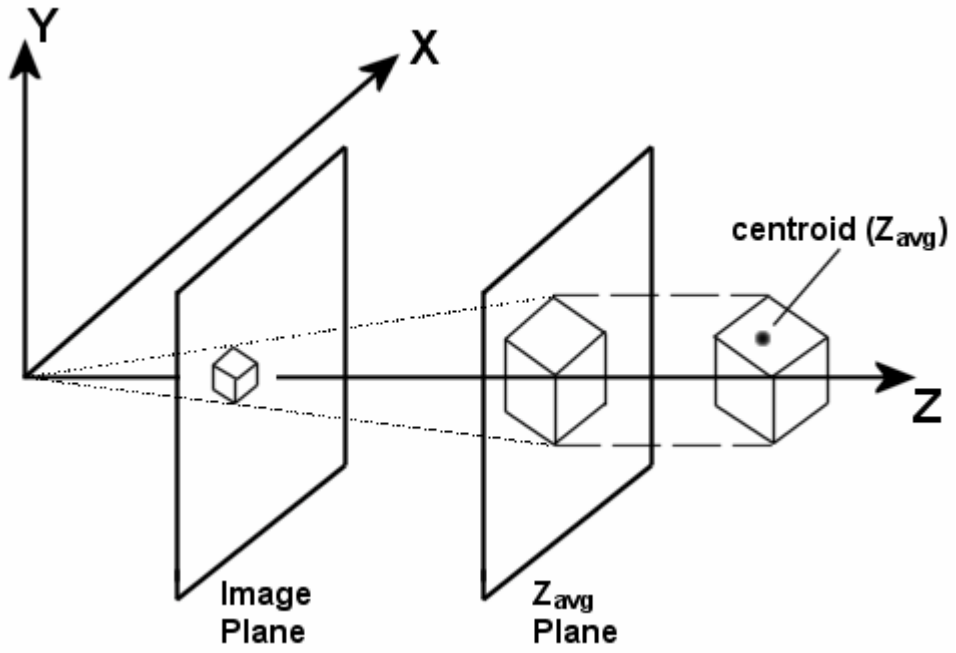


Figure 2.3 – The Weak-perspective Camera Model

2.1.3 Camera Parameters

There are two subsets of camera parameters that can be used to determine the relationship between coordinate systems. Known as the *intrinsic* and *extrinsic* parameters in the computer vision field, they are defined as follows:

Intrinsic Camera Parameters

The intrinsic parameters are those related to the internal geometry of a physical camera. In other words, they represent the optical, geometric, and digital characteristics of a camera. The parameters are:

1) The focal length
2) The location of the image center in pixel space
3) The pixel size in the horizontal and vertical directions (see [TRUC98] for further details related to image sensors and pixel aspect ratios)
4) The coefficient to account for radial distortion from the optics

The first parameter, the focal length, is the same parameter described in Section 2.1.2.

The second and third parameters allow us to link image coordinates (x_{im}, y_{im}), in pixels, with the respective coordinates (x, y) in the camera coordinate system. This is done quite simply:

$$x = -(x_{im} - o_x)s_x$$
$$y = -(y_{im} - o_y)s_y$$

Equation 2.3 – Linking pixel coordinates and camera coordinates

where (o_x, o_y) define the pixel coordinates of the *principal point*, and (s_x, s_y) define the size of the pixels (in millimeters), in the horizontal and vertical directions respectively. Using Figure 2.2 as our reference, the sign change is required if we assume that the image has its x coordinates increasing to the right, and the y coordinate increasing going down, with the origin of the image in the top-left corner.

The final parameter allows us to account for *radial distortions* that are evident when using camera optics with large fields of view [TRUC98]. Typically, the distortions are most pronounced at the periphery of the image, and thus can be corrected using a simple radial displacement of the form

$$x = x_d(1 + k_1 r^2 + k_2 r^4)$$
$$y = y_d(1 + k_1 r^2 + k_2 r^4)$$

Equation 2.4 – Correcting for radial distortion

where (x_d, y_d) is the distorted point in camera space, and $r^2 = x_d^2 + y_d^2$. k_1 and k_2 are additional intrinsic camera parameters, where $k_2 << k_1$. Usually k_2 is set to 0. In many cases, radial distortion can be ignored unless very high accuracy is required in all parts of the image.

Extrinsic Camera Parameters

The extrinsic parameters are concerned with external properties of a camera, such as position and orientation information. They uniquely identify the transformation between the unknown camera coordinate system and the known world coordinate system [TRUC98]. The parameters, as depicted in Figure 2.4, are:

1) The 3x3 rotation matrix *R* that brings the corresponding axes of the two coordinate systems onto one another
2) The 3D translation vector **T** describing the relative positions of the origins of the two coordinate systems

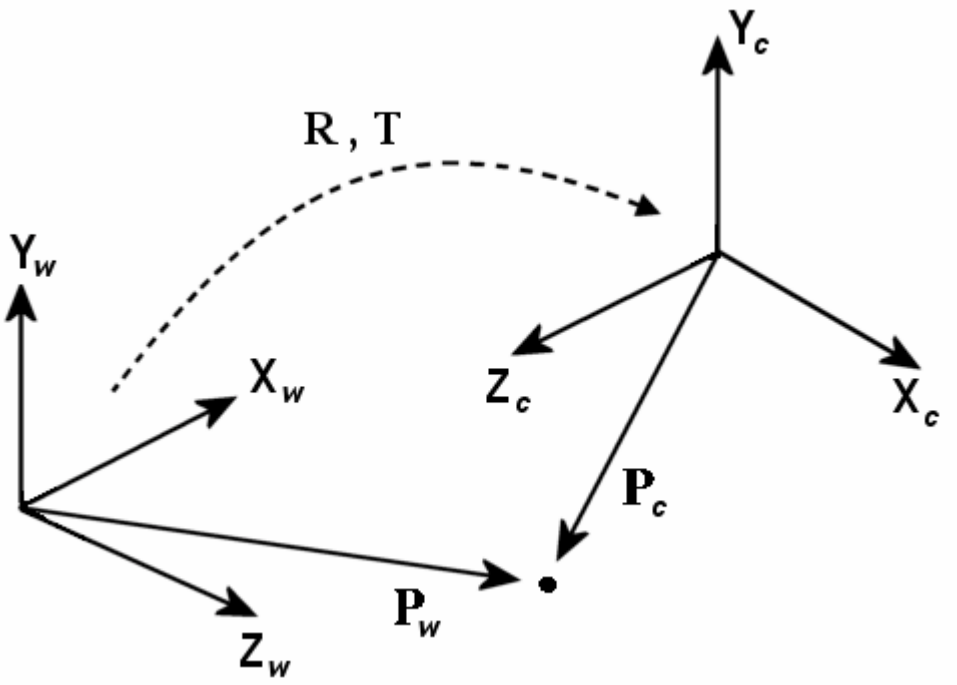


Figure 2.4 – Transformation between camera and world coordinate systems [TRUC98]

In other words, if we have a point $\mathbf{P_w}$ in world coordinates, then the same point in camera coordinates, $\mathbf{P_c}$, would be:

$$\mathbf{P_c} = R\mathbf{P_w} + \mathbf{T}$$

Equation 2.5 – Transformation from world to camera coordinates

where

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

defines the rotational information.

Therefore, if we neglect radial distortions, we can plug Equation 2.3 and Equation 2.5 into our perspective projection equation (Equation 2.1), resulting in:

$$-(x_{im} - o_x)s_x = f\frac{\mathbf{R_1}^{\mathrm{T}}(\mathbf{P_w} - \mathbf{T})}{\mathbf{R_3}^{\mathrm{T}}(\mathbf{P_w} - \mathbf{T})}$$

$$-(y_{im} - o_y)s_y = f\frac{\mathbf{R_2}^{\mathrm{T}}(\mathbf{P_w} - \mathbf{T})}{\mathbf{R_3}^{\mathrm{T}}(\mathbf{P_w} - \mathbf{T})}$$

Equation 2.6 – Relation between 3D world coordinates and the corresponding image coordinates, using intrinsic and extrinsic parameters.

where $\mathbf{R_i}$, i = 1, 2, 3, denotes the 3D vector formed by the i-th row of the matrix R.

Separating the intrinsic and extrinsic components, and placing the equations into matrix form, we get:

$$M_{int} = \begin{pmatrix} f_u & 0 & o_x \\ 0 & f_v & o_y \\ 0 & 0 & 1 \end{pmatrix}$$

Equation 2.7 – Intrinsic camera parameter matrix

where $f_u = -f / s_x$ and $f_v = -f / s_y$, which defines the transformation between camera space and image space, and

$$M_{ext} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix}$$

Equation 2.8 – Extrinsic camera parameter matrix

where $t_1 = -\mathbf{R_1}^T\mathbf{T}$, $t_2 = -\mathbf{R_2}^T\mathbf{T}$, and $t_3 = -\mathbf{R_3}^T\mathbf{T}$, which defines the transformation between world coordinates and camera coordinates.

Therefore, our projection equation can now be expressed in homogeneous matrix form:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = M_{int} M_{ext} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$$

Equation 2.9 – Projection equation in homogeneous matrix form

where $x_1/x_3 = x_{im}$ and $x_2/x_3 = y_{im}$.

Going back to our camera models, and setting some reasonable constraints on our parameters ($o_x = 0$, $o_y = 0$), we can express the *perspective projection matrix* as simply:

$$M = M_{int} M_{ext} = \begin{pmatrix} f_u r_{11} & f_u r_{12} & f_u r_{13} & f_u t_1 \\ f_v r_{21} & f_v r_{22} & f_v r_{23} & f_v t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix}$$

Equation 2.10 – Perspective projection matrix

Similarly, [TRUC98] defines the *weak-perspective camera matrix* as:

$$M_{wp} = M_{int} M_{ext} = \begin{pmatrix} f_u r_{11} & f_u r_{12} & f_u r_{13} & f_u t_1 \\ f_v r_{21} & f_v r_{22} & f_v r_{23} & f_v t_2 \\ 0 & 0 & 0 & \mathbf{R_3}^T\mathbf{(P'-T)} \end{pmatrix}$$

Equation 2.11 – Weak-perspective camera matrix

where $\mathbf{P'}$ is the centroid of two points, P_1 and P_2 in 3D space.

2.1.4 Camera Calibration

Now that we have defined our camera models and camera parameters, we now have a method to associate the various coordinate systems from Figure 2.1. However, this

assumes that we know the actual values of our intrinsic and extrinsic parameters. The process of determining the intrinsic and extrinsic camera parameters is known as the *camera calibration* problem [TRUC98].

The basic idea is to solve for the camera parameters based on the projection equations of known 3D coordinates and their associated 2D projections. Six or more such correspondences are required in order to solve a linear system of equations that can recover the twelve elements of a 3x4 projection matrix [ZISS98]. There are two common methods for camera calibration [TRUC98]. The first method attempts to directly estimate the intrinsic and extrinsic parameters based on finding features in a known calibration pattern, as depicted in Figure 2.5. The second method first attempts to estimate the projection matrix linking world and image coordinates, and then uses the entries of this matrix to solve for the camera parameters.

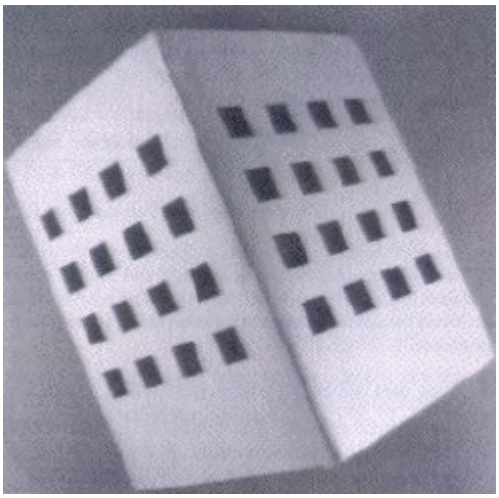


Figure 2.5 – A camera calibration pattern [TRUC98]

The major difficulty with these calibration approaches is the need to perform them manually in a separate calibration procedure. For the purposes of augmented reality, efficient and accurate camera calibration remains an open problem. Section 2.4 and 2.6 discuss some proposed calibration solutions with respect to vision-based augmented reality systems.

2.2 Registration Errors

Since all augmented reality systems experience registration errors from many sources, it is worth describing them briefly since they affect the performance and immersion of the augmented environment. The two main categories of registration errors, as described by [AZUM97a] are *static* and *dynamic*.

2.2.1 Static Errors

Static registration errors are defined as errors that occur when nothing is moving in the scene. An example of a static error would be a virtual object appearing in different places when viewed from different viewpoints.

There are four main sources of static errors [AZUM97a]:

- Optical distortion: Radial distortions, as described briefly in Section 2.1.3, can cause registration errors when using optical see-through displays. These errors can be compensated via additional optics, or by using image-warping techniques.
- Tracking errors: Discrepancies in the reported location of a user's head or other objects in a scene are the most serious source of static registration errors, since they are difficult to measure or eliminate.
- Mechanical misalignments: Hardware components (cameras, optics, monitors) that aren't particularly rigid may experience subtle changes as a user's position or orientation changes. The best solution to this static error source is to build hardware properly from the very beginning, since mechanical problems are difficult to compensate for.
- Incorrect viewing parameters: These include discrepancies in the parameters used to render the virtual objects into the scene, such as the offset between the head tracker and a user's eyes, the field of view, and the center of projection. Accurate camera-calibration techniques help to reduce the effects of these static errors.

2.2.2 Dynamic Errors

Dynamic registration errors occur when either the user or virtual object are in motion, resulting in slight positional shifts of the augmented entities [VALL98]. In other words, the virtual objects seem to be out of sync with the real-world scene.

Most dynamic errors occur as a result of system delays [AZUM97a]. For example, a vision-based augmented reality system must typically capture an image from the scene, determine the camera pose, transform virtual objects into camera space, render the virtual objects onto the original image, and display the final augmented scene onto the display. Each of these phases of the augmentation system requires a certain amount of processing time. Suppose the total required time is s. If an image is captured at time t, then the user will not see the final image until time $t + s$. If $s > 50$ms, a user's sense of immersion in an augmented environment is strongly undermined [AZUM97a].

While constant technological improvements in processor performance will help resolve these system delays, they will never disappear completely. Therefore, other techniques must be employed in order to reduce the effects of lag, such as viewpoint prediction, temporal stream matching (eg. video see-through displays), or more efficient algorithms in each phase of the augmentation system [AZUM97a].

2.3 Sensor-based Registration Approaches

In the past, the majority of augmented reality systems have relied on sensors, such as magnetic, mechanical, or inertial, in order to measure the pose of the camera. Note that, due to the high-precision requirements of the registration process, no sensing device alone has achieved the results required for most augmented reality applications [VALL98].

2.3.1 Magnetic Sensors

Magnetic tracking systems use digital compasses and the earth's magnetic field to estimate the pose of the camera with respect to some reference position and orientation. The major advantage of these sensors is their portability; they can be used to perform augmentations almost anywhere. Unfortunately, magnetic sensors are easily disturbed by the presence of metallic objects in the environment. Additionally, these sensors experience latencies that can only be corrected via prediction algorithms. Uncalibrated magnetic trackers can exhibit errors of 10cm or more, while carefully calibrated systems can reduce measurement errors to within 2cm [STAT00].

2.3.2 Inertial Sensors

Some basic inertial tracking techniques are discussed in [YOU99a]. Inertial sensors consist of two devices:

- Gyroscopes: provide a rate of rotation that can be used to determine orientation changes of a user's head.
- Accelerometers: measure the linear acceleration of a user's head to determine translations.

While head pose can thus be found rather quickly, inertial sensors suffer from accumulation errors over time that can adversely affect registration stability.

2.4 Vision-based Registration Approaches

In comparison to sensor-based techniques, vision-based techniques have experienced significantly better results when attempting to achieve the high-precision requirements of the registration process. This section describes some of the proposed solutions, along with some of the limitations and challenges that must be overcome.

2.4.1 Tracking Known Patterns

The pattern-based ARToolkit technology developed at the University of Washington, which can augment virtual objects onto predetermined black and white square markers, is used in [BILL01a, BILL01b, KATO00a, KATO00b, REGE01]. The technology first thresholds the captured frame into a binary image. Black regions that can be fitted by four lines are then normalized so that they are front facing and square. These normalized images are then matched against known pattern templates, and if a match is found the region is marked as being identified. Once identified, the position and orientation of the pattern is determined with respect to the camera. Once the pose has been determined, any virtual objects associated with the identified pattern are augmented into either the original video frame or optically in an HMD. Figure 2.6 shows some examples of the ARToolkit technology.

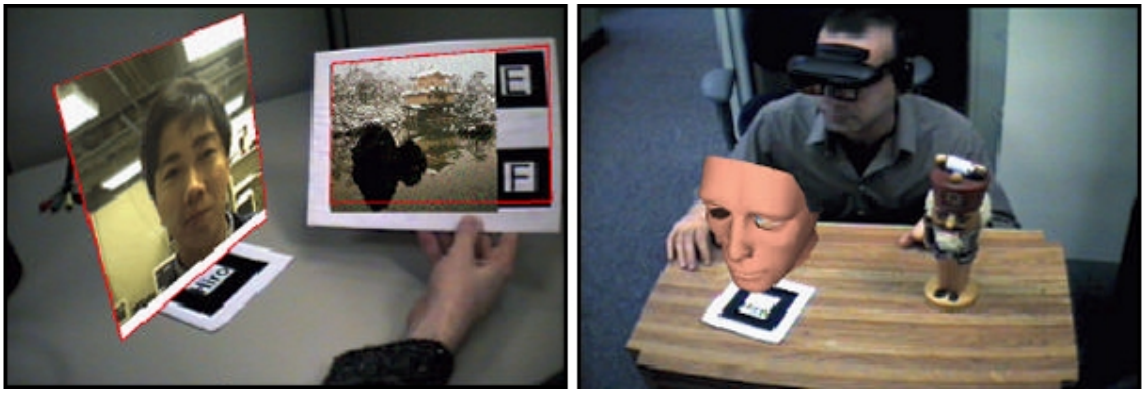


Figure 2.6 – ARToolkit augmentation [KATO00a].

The advantage of the ARToolkit technology is its simplicity and performance. It can operate at interactive frame rates on consumer-level PCs using off-the-shelf USB camera hardware. Fast movements do not pose any major problems due to the global image processing performed on each frame. However, the simplicity comes at the expense of robustness. Any occlusion of any part of the pattern in the video frame results in a loss of augmentation (since the line-fitting process fails). Therefore, attempting to view augmented objects close-up is not possible since portions of the marker may be out of view. Further, any attempts at manipulating the virtual objects (via hand gesture recognition or haptic input devices) would also not be possible since the manipulation

device would occlude the pattern. Additionally, intrinsic calibration is considered to be a separate step that must be performed before a particular camera can be used with the system. Only extrinsic (pose) parameters are determined at run-time based on the orientation of the pattern in the video frame.

Similar to the ARToolkit technology, [REKI98] tracks black and white 2D matrix markers that can be attached to real-world objects, onto which text or virtual 3D objects can be augmented. A similar normalization/unwarping process is employed in order to recognize the pattern in the video image. Unlike ARToolkit, however, the choice of pattern is restricted to a square shaped barcode arrangement. This allows the augmentation system to recognize 2^{16} unique patterns onto which different objects can be overlaid. Figure 2.7 shows an example of a 2D matrix marker and its unwarped equivalent from a video frame.

Figure 2.7 – (a) Original 2D matrix pattern, (b) 2D matrix restored from video [REKI98]

Patterns consisting of black and white circular patches of dots, where the dots are arranged along a major and minor axis, are tracked in [MOLI01]. By restricting the arrangements, patterns are unambiguously detected and pose computations are simplified. Valid dot arrangements are determined by treating them as vertices in a graph and finding a minimum spanning tree between them. While coloured dots are useful for pattern self-identification, they are adversely affected by changes in lighting, camera quality, and dot orientation. Pose is estimated in an orthoperspective (similar to weak-perspective) space by using three successfully tracked non-collinear points.

Their implementation achieves interactive frame rates (15-30 Hz), and can handle brief partial occlusion of the markers by predicting motion via Kalman filtering. However, while circular patch finding is relatively efficient, the centroid computation of circular

patches can vary from frame to frame due to image noise. Thus this method may suffer from significant jitter. Additionally, the authors note that approximation of perspective using an orthoperspective projection can experience up to 10 degrees of angular error when viewing objects at close range.

In a similar fashion, [VALL98] uses an affine coordinate system which doesn't require any Euclidean calibration of the video camera to perform augmentations. Rectangular blobs arranged on two orthogonal planar patterns are tracked in real-time, and lines are fitted to these regions to extract four corner points. These four corners are then used to create an affine projection matrix that can be directly used to augment virtual objects onto the pattern.

Similar to the ARToolkit method, this approach does not offer any robustness to occlusion; no affine projection matrix is computed if the system detects less than the required number of features. Additionally, the affine augmentations are only realistic when the depth difference between the nearest and farthest feature points is small compared to the focal length of the camera lens.

2.4.2 Tracking Natural Features

A system that tracks markerless planar regions in natural environments is described in [SIMO00]. This is a special case which works well in indoor environments consisting of textured ceiling tiles or floors, as well as in outdoor environments with rough ground textures (such as grass). Additionally, their approach can "hand off" tracking from one plane to another, such that the same plane need not be visible throughout the video sequence. The key to their approach involves robustly computing a homography (a 2D-2D transformation) between feature points from one frame to the next. The homography can then be used to extract extrinsic calibration parameters since the planar region imposes a natural frame of reference. Further, they describe an estimation scheme for the intrinsic parameters if they are unknown.

Unfortunately, this approach requires the detection of hundreds of corner features in every frame of video, as well as correlating these features between frames in order to find feature correspondences for the homography computation. Thus the tracking is extremely robust to noise and occlusion, but the overall approach is not suited for real-time performance on current consumer-level hardware.

Motion is tracked in natural environments by estimating optical flow of both feature points and entire regions in [NEUM99, YOU99b]. Their approach is used in combination with inertial and vision-based trackers in order to increase precision. Unfortunately, the algorithm currently does not operate in real-time and requires a separate intrinsic camera calibration step.

A stereo camera configuration is used in [KANB01] in order to track natural features for augmentation purposes. Their system first estimates a projection matrix by viewing a set of pre-defined fiducial markers. The projection matrix is then updated as a user moves around in the environment by picking up on natural features (such as corners or edges) that can be easily matched between the stereo (left/right) images. The use of stereo is both an advantage and disadvantage. On the one hand, epipolar constraints [XU96] can be used to determine whether feature points in the left/right images correspond, and the 3D positions of the feature points can be computed using stereo triangulation techniques. On the other hand, their stereo camera configuration requires expensive hardware, and the cameras require performing a separate intrinsic camera calibration procedure.

2.5 Hybrid Registration Approaches

Some researchers have concluded that sensor-based techniques alone will never work well enough to be the only tracking technology for augmented reality, due to the inaccuracies and latencies they exhibit. Therefore, the most promising approach may be to use sensors to grossly estimate the pose, followed by vision-based techniques to fine-tune the registration process [VALL98]. The idea is to combine the strengths of multiple approaches in order to overcome the respective weaknesses.

2.5.1 Magnetic and Vision

Optical and magnetic tracking for the Studierstube project is discussed in [AUER99, AUER00]. The system is able to track the corners of black rectangles on a white background, the locations of which have been determined in a prior calibration procedure. The magnetic tracker is then used to roughly estimate the positions of these corner features. The predicted corner locations are then given to the optical tracker, which places a 5x5 subwindow around the predicted feature location. After performing subpixel corner and edge detection as defined in [BRAN99], the final position of the corner feature is determined. Camera pose is then determined based on the 2D positions of the features in image space and the known 3D positions in the world coordinate space. Note that the intrinsic camera parameters are determined in a separate calibration procedure.

The system performs at interactive frame rates, with position jitter between 0.4mm to 2.7mm. Since the pose estimation only requires four successfully tracked feature points, the system is also robust to partial occlusions of the tracked pattern.

In a similar fashion, [STAT00] combines magnetic and vision-based tracking in order to achieve robust 3D augmentations. The vision-based tracker first attempts to locate two-colour circular blobs consisting of an inner circle and outer ring, where the diameter of the outer ring is three times larger than the diameter of the inner blob. Four fluorescent colours (red, green, blue, yellow) are used, resulting in a maximum of twelve possible unique landmarks. When three or more non-collinear landmarks are visible, the computed centroid positions are used to determine the pose of the camera. Intrinsic camera parameters, however, are determined in an independent calibration procedure. The computed head pose is then fed to the magnetic tracker in order to determine the amount of error in the magnetic tracker's estimated pose. Assuming temporal coherency, this error amount is then used to predict head pose in the next frame. Thus in the next

frame, the predicted head pose from the magnetic tracker is fed into the vision-based tracker in order to find more landmarks in local search areas.

Compared to single-colour blob finders, the use of two-coloured landmarks makes the system robust to invalid marker detection. Additionally, blob finding in general is much quicker than complicated edge or corner finding algorithms. Further, the use of two-coloured blobs makes the system robust to partial occlusions since the entire blob position and size can be estimated with only a portion of a blob visible.

The prediction mechanism allows the augmentation system to continue functioning even if some or all of the landmarks are briefly occluded, as opposed to breaking down completely. Experimental results show that position and orientation errors of this hybrid tracker are typically less than 2mm and 0.2°.

The major disadvantage with this blob scheme is the limitation to twelve unique patterns. Scalability is a concern since increasing the number of uniquely detectable colours is a difficult task due to colour thresholding limitations and varying camera qualities. For example, trying to detect an orange blob would cause problems with red and yellow blob detection since the RGB components are similar.

The ARQuake project described in [THOM00] uses a combination of magnetic tracking, a global position system (GPS), and vision-based marker tracking in order to present an augmented reality version of the popular Quake video game. The GPS and magnetic tracking system is used for augmentations at large outdoor distances from the user (> 50m), since GPS precision is rather coarse for close-up augmentations. For closer augmentations (such as when navigating indoor environments), the system makes use of the ARToolkit library [KATO00a]. Throughout the indoor environment, variable sized black and white patterns (ranging from 19cm x 19cm to 1m x 1m) are placed in strategic locations in order to recreate a common coordinate system for the Quake 3D world.

The AR^{2}Hockey augmented reality air hockey game [OHSH98] also uses a hybrid magnetic and vision system for tracking purposes. Similar to the previously mentioned approaches, the magnetic sensor is used to predict user motion while the vision system tracks visual landmarks in order to refine the estimated pose.

2.5.2 Inertial and Vision

An inertial and vision augmentation system is described in [YOU99a]. Inertial data is used to determine approximate 2D feature motion, and vision-based feature tracking then refines these estimates by performing local searches in order to determine the true feature positions. In this way, the gyro data increases the robustness and performance of the vision-based tracker, while the vision system corrects for the accumulated drift of the inertial system. A similar system is described in [KANB01], but using a stereo camera configuration instead of a single camera in order to improve robustness.

2.6 Calibrated vs. Uncalibrated Registration

The majority of augmentation systems described in the previous sections relied on manual calibration procedures to determine the intrinsic camera parameters, followed by various 3-point [MOLI01, STAT00] or 4-point [OBER93, STUR00] pose estimation techniques to determine the extrinsic parameters.

The problem with manual intrinsic calibration is the lack of support for zoom lenses, since the focal length changes. To address this problem, [SIMO99] proposes a method to detect camera motions and zoom variations in a video sequence (between two consecutive frames). Assuming that zoom and camera motion do not occur in the same frame, their algorithm is able to perform precise registrations in each separate case. If camera motion is detected, the system assumes the focal length is constant and thus can use a 3-point or 4-point pose estimation algorithm for the extrinsic parameters. On the other hand, if zoom is detected, the system only needs to determine new intrinsic parameters based on the positional change of tracked 2D/3D feature correspondences. Of

course, this assumes that the initial intrinsic parameters are known at startup time. Additionally, since focal length will be progressively adjusted during zoom detections, there is the potential for accumulation error.

Recently, computer vision researchers have been experimenting with semi-automatic calibration techniques that can be exploited by augmented reality systems. An algorithm that can recover both intrinsic and extrinsic parameters by tracking known quadrangular targets is described in [ABID00]. A semi-automatic technique that can recover camera parameters from a homography by tracking a known planar pattern is also described in [ZHAN00]. Similarly, [SIMO00] uses a homography to estimate the intrinsic and extrinsic parameters when tracking planar structures in natural environments.

Some researchers have also been experimenting with completely uncalibrated registration for augmented reality. Affine object representations for a real-time augmentation system are described in [KUTU98, VALL98], and thus do not require an explicit Euclidean calibration of the camera. Therefore, virtual objects can be registered by directly applying a computed 3x4 orthographic projection matrix. As described earlier, the disadvantage of this approach is the lack of realistic perspective distortion on the virtual objects when objects are observed close-up. Additionally, the lack of a proper perspective space limits the systems ability to accurately handle traditional computer graphics effects such as lighting and texture mapping on the virtual objects.

Building upon the work in [KUTU98], [SEO00] presents an algorithm for computing a perspective projection matrix without explicit Euclidean camera calibration. The technique is based upon projective reconstruction, which involves determining the fundamental matrix [XU96] between two images in a video sequence in order to reconstruct the 3D position of tracked 2D feature points. The downfall with this approach is the time-consuming fundamental matrix computation that occurs between every pair of consecutive video frames.

For some applications, intrinsic and extrinsic camera calibration may not be required at all. Consider annotating real-world objects with simple 2D text or graphics. In these cases, accurate 2D tracking of planar patterns would be sufficient since a homography would precisely define a mapping from the 2D pattern space to the video frame, with automatic support for zoom lenses.

2.7 Open Problems

While a variety of solutions to the registration problem have been proposed in the augmented reality literature, none fully address all the requirements of a robust system that is ready for consumer-level products. Some systems provide excellent stability and robustness of registration, but require sophisticated calibration steps and expensive hybrid tracking equipment. Other accurate vision-only registration approaches work with low-cost hardware, but the computational costs currently don't allow interactive frame rates. Still others achieve real-time performance on low-cost hardware, but at the expense of stability and robustness. As can be seen, the various approaches all establish their own balance between cost, performance, and accuracy based on application-specific requirements.

The ultimate goal of augmented reality is to provide seamless integration of virtual objects with natural, unprepared environments. Additionally, this integration should be automatic and reliable in a wide variety of lighting conditions and at various distances and user orientations, with the ability to interact with the virtual imagery. This requires solutions to many open problems, particularly in automatic calibration systems and tracking in arbitrary environments [AZUM01]. No system has addressed all of the requirements of the ideal augmented reality environment, but progress is being made in each of the problem domains. Clearly, an evolutionary rather than revolutionary approach is required to bring AR out from the research labs and into mainstream products.

Chapter 3

Planar Homographies for Registration

This chapter discusses planar homographies and how they relate to the registration problem. A basic overview of the mathematics behind homographies is first provided, followed by a description of how they can be used for 2D augmentations. An existing method to estimate all intrinsic and extrinsic camera parameters from a homography is then described, which allows our 3D augmentation system to autocalibrate itself.

3.1 Mathematical Background

For pattern-based augmented reality, a planar pattern defines a world coordinate system into which virtual objects will be placed, as depicted in Figure 3.1. It would be convenient if the planar pattern itself could be used to determine a projection matrix that could be directly applied to the coordinates of a virtual object for augmentation purposes. This would eliminate the need for a separate complicated calibration procedure, thus simplifying the system for the end-user.

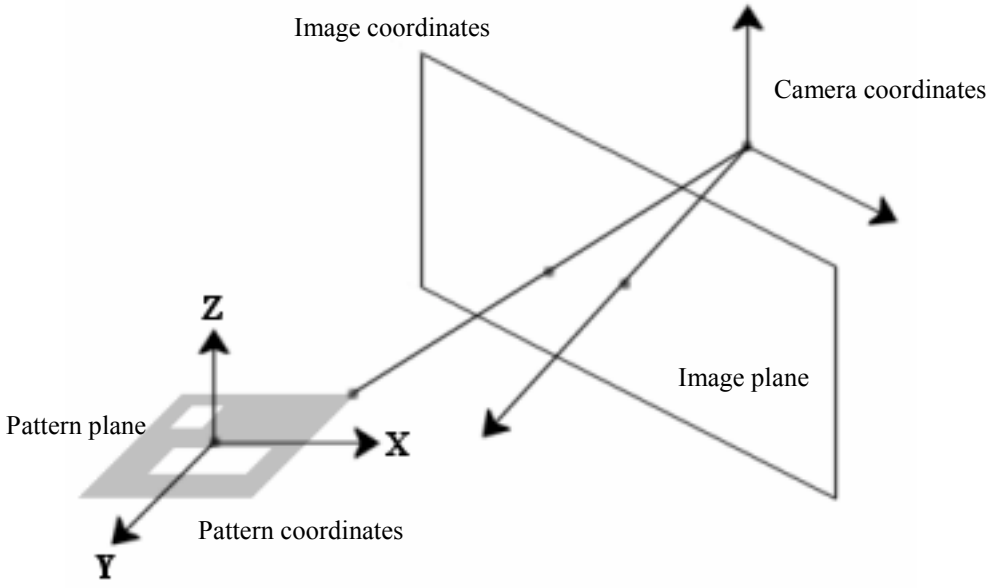


Figure 3.1 – World coordinate system induced by a planar pattern

If we make the assumption that the planar pattern defines the Z=0 plane in world space, we note the following simplification when projecting points on the planar pattern into image space

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ 1 \end{pmatrix} = H \begin{pmatrix} x_w \\ y_w \\ 1 \end{pmatrix}$$

where p_{ij} defines the i,j-th element of the perspective projection matrix described earlier, and H is a 3x3 2D-to-2D projective transformation known as a homography [ZISS00].

In other words, we have

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \lambda H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Equation 3.1 – 2D-to-2D projective transformation

where $\mathbf{x} = (x, y)$ is a point in the original plane, $\mathbf{x}' = (x', y')$ is the corresponding point in the image, and λ is a scale factor.

We can rewrite the equation as

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$
$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

where h_{ij} defines the i,j-th element of H. This can be further rewritten as two linear equations

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}$$
$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}$$

In matrix form we have

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{pmatrix} \mathbf{h} = \mathbf{0}$$

where $\mathbf{h} = [h_{11}\ h_{12}\ h_{13}\ h_{21}\ h_{22}\ h_{23}\ h_{31}\ h_{32}\ h_{33}]^T$ is a 9-element vector containing the elements of H.

Therefore with four such non-collinear point correspondences, we can solve for all the elements of H as follows

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2'x_2 & -y_2'y_2 & -y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3'x_3 & -x_3'y_3 & -x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y_3'x_3 & -y_3'y_3 & -y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4'x_4 & -x_4'y_4 & -x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y_4'x_4 & -y_4'y_4 & -y_4' \end{pmatrix} \mathbf{h} = A\mathbf{h} = \mathbf{0}$$

The solution **h** is thus the null-space of the 8x9 matrix A, which can be solved using known methods such as singular value decomposition [TRUC98].

3.2 Uncalibrated 2D Augmentations

Assuming that four pattern to image point correspondences can be made (so that H can be computed), augmentations of 2D objects onto the plane are possible. Equation 3.1 describes the required mapping that needs to be made in order to convert 2D locations on the planar pattern into the appropriate 2D locations in a frame of video.

Therefore, given a texture-mapped 3D polygon defined relative to the plane, with Z=0 for all vertices, the polygon could be transformed into the image with proper perspective distortion. Virtual video screens, 2D board games, and textual overlays can be achieved automatically without the need for a manual calibration procedure. Figure 3.2 shows an example of a 2D image being augmented onto a planar pattern.

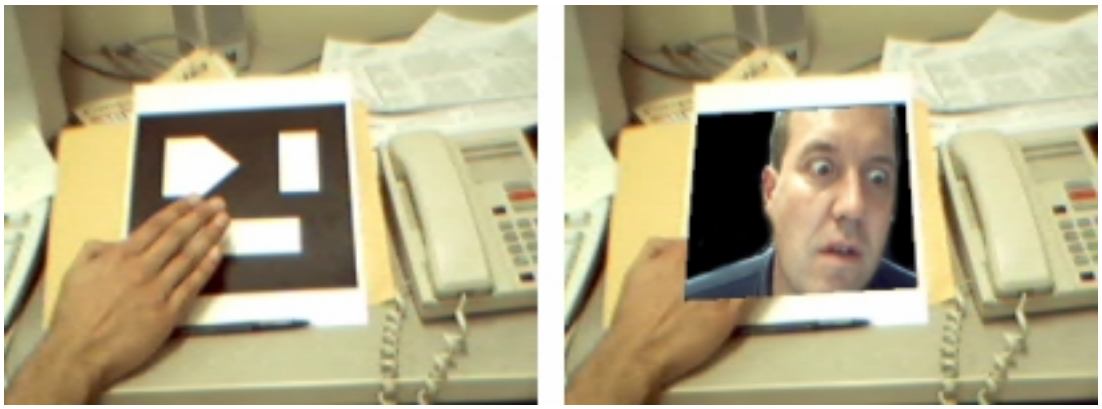


Figure 3.2 – Uncalibrated registration of a 2D video image

3.3 Camera Parameter Estimation and 3D Augmentations

Unfortunately, *H* alone cannot be directly used to augment virtual 3D objects into the image, since the Z component from pattern space is assumed to always be zero. However, recent work in [SHIM98, ZHAN00] involving planar homographies and camera calibration can be used to recover the missing camera parameters.

Since *H* is a simplification of the general perspective projection matrix from Equation 2.10 (where Z=0), it is clear that *H* can be defined as

$$H = \begin{pmatrix} f_u r_{11} & f_u r_{12} & f_u t_1 \\ f_v r_{21} & f_v r_{22} & f_v t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix}$$

Equation 3.2 – *H* as a simplified projection matrix

Since *R* is a rotation matrix, its orthogonality properties give

$$r_{11}^{\ 2} + r_{21}^{\ 2} + r_{31}^{\ 2} = 1 \quad \textbf{(Equation 3.3)}$$

$$r_{12}^{\ 2} + r_{22}^{\ 2} + r_{32}^{\ 2} = 1 \quad \textbf{(Equation 3.4)}$$

$$r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} = 0 \quad \textbf{(Equation 3.5)}$$

Combining Equation 3.5 with the representation of *H* in Equation 3.2 above results in

$$h_{11}h_{12} / f_u^{\,2} + h_{21}h_{22} / f_v^{\,2} + h_{31}h_{32} = 0 \quad \textbf{(Equation 3.6)}$$

Similarly, Equations 3.3 and 3.4 provide

$$\lambda^2 (h_{11}^2 / f_u^2 + h_{21}^2 / f_v^2 + h_{31}^2) = 1 \quad \textbf{(Equation 3.7)}$$

$$\lambda^2 (h_{12}^2 / f_u^2 + h_{22}^2 / f_v^2 + h_{32}^2) = 1 \quad \textbf{(Equation 3.8)}$$

By eliminating λ^2 in Equations 3.7 and 3.8, we get

$$(h_{11}^2 - h_{12}^2) / f_u^2 + (h_{21}^2 - h_{22}^2) / f_v^2 + h_{31}^2 - h_{32}^2 = 0 \quad \textbf{(Equation 3.9)}$$

Using Equations 3.6 and 3.9, we can solve for both f_u and f_v

$$f_u = \sqrt{\frac{h_{11}h_{12}(h_{21}^2 - h_{22}^2) - h_{21}h_{22}(h_{11}^2 - h_{12}^2)}{-h_{31}h_{32}(h_{21}^2 - h_{22}^2) + h_{21}h_{22}(h_{31}^2 - h_{32}^2)}} \quad \textbf{(Equation 3.10)}$$

$$f_v = \sqrt{\frac{h_{11}h_{12}(h_{21}^2 - h_{22}^2) - h_{21}h_{22}(h_{11}^2 - h_{12}^2)}{-h_{31}h_{32}(h_{11}^2 - h_{12}^2) + h_{11}h_{12}(h_{31}^2 - h_{32}^2)}} \quad \textbf{(Equation 3.11)}$$

Once the intrinsic parameters (the focal lengths) have been computed, Equation 3.7 or Equation 3.8 can be used to compute the value for λ. For example, using Equation 3.7 we have

$$\lambda = \frac{1}{\sqrt{h_{11}^2 / f_u^2 + h_{21}^2 / f_v^2 + h_{31}^2}} \quad \textbf{(Equation 3.12)}$$

All of the extrinsic camera parameters (Equation 2.8) can then be computed as follows

$r_{11} = \lambda h_{11} / f_u$	$r_{12} = \lambda h_{12} / f_u$	$r_{13} = r_{21}r_{32} - r_{31}r_{22}$	$t_1 = \lambda h_{13} / f_u$
$r_{21} = \lambda h_{21} / f_v$	$r_{22} = \lambda h_{22} / f_v$	$r_{23} = r_{31}r_{12} - r_{11}r_{32}$	$t_2 = \lambda h_{23} / f_v$
$r_{31} = \lambda h_{31}$	$r_{32} = \lambda h_{32}$	$r_{33} = r_{11}r_{22} - r_{21}r_{12}$	$t_3 = \lambda h_{33}$

Note that r_{13}, r_{23}, and r_{33} are found by using the fact that the Z axis of the rotation matrix is orthogonal to the other two.

As a result of recovering both the intrinsic and extrinsic camera parameters, virtual 3D objects can be augmented onto the planar pattern in image space, under full perspective, without the need for a separate manual calibration step (see Figure 3.3). In other words, the homography can be used to autocalibrate the camera, which provides us with a full perspective projection matrix. Once we have the camera projection matrix we can then use it to render, in a perspective-correct manner, the 3D coordinates of any virtual 3D object that is defined in pattern space (i.e. relative to the centre of the pattern in Figure 3.1). Figure 3.4 shows some additional examples of the 3D augmentation. Note that we did not need to know anything about the camera parameters a-priori to compute the homography, or to compute the projection matrix. All that is required is a method to determine reliable correspondences between pattern space and image space in a real-time video sequence. This is the focus of Chapter 4.

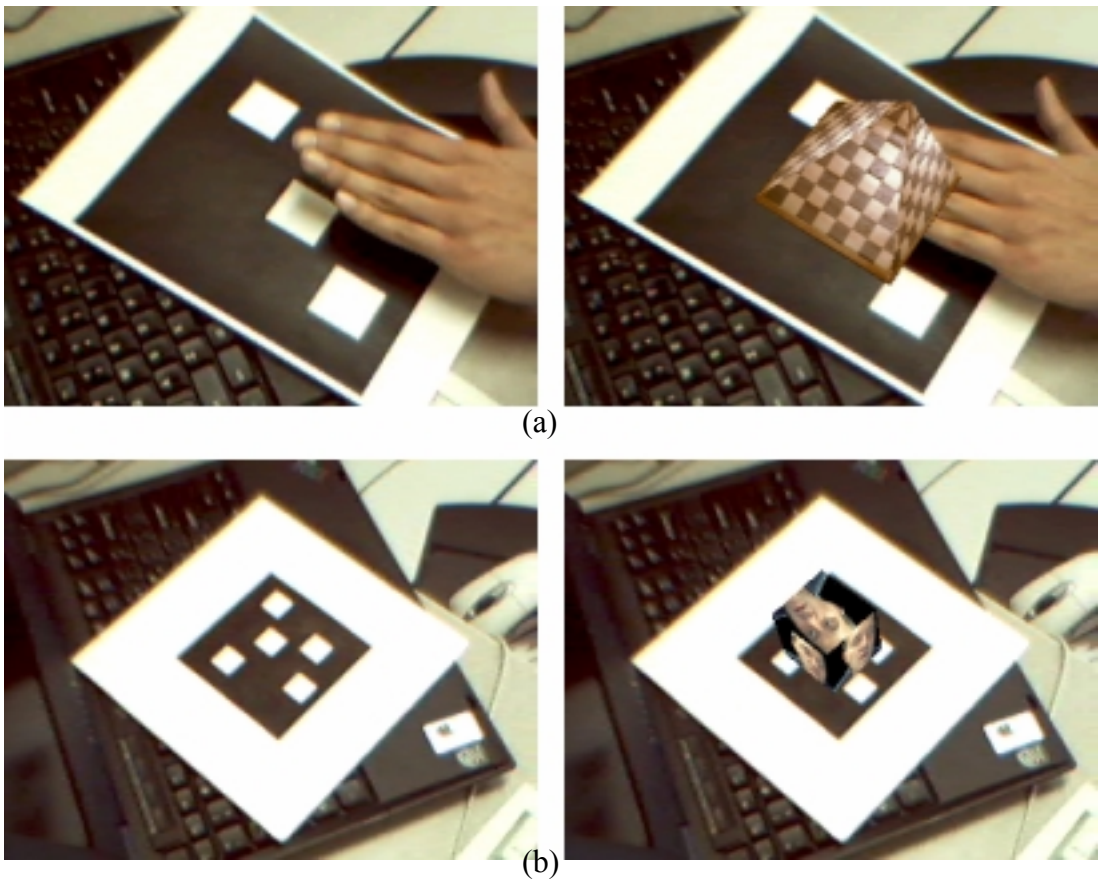


Figure 3.3 – Automatically calibrated 3D augmentation examples (a) virtual pyramid, (b) virtual cube.

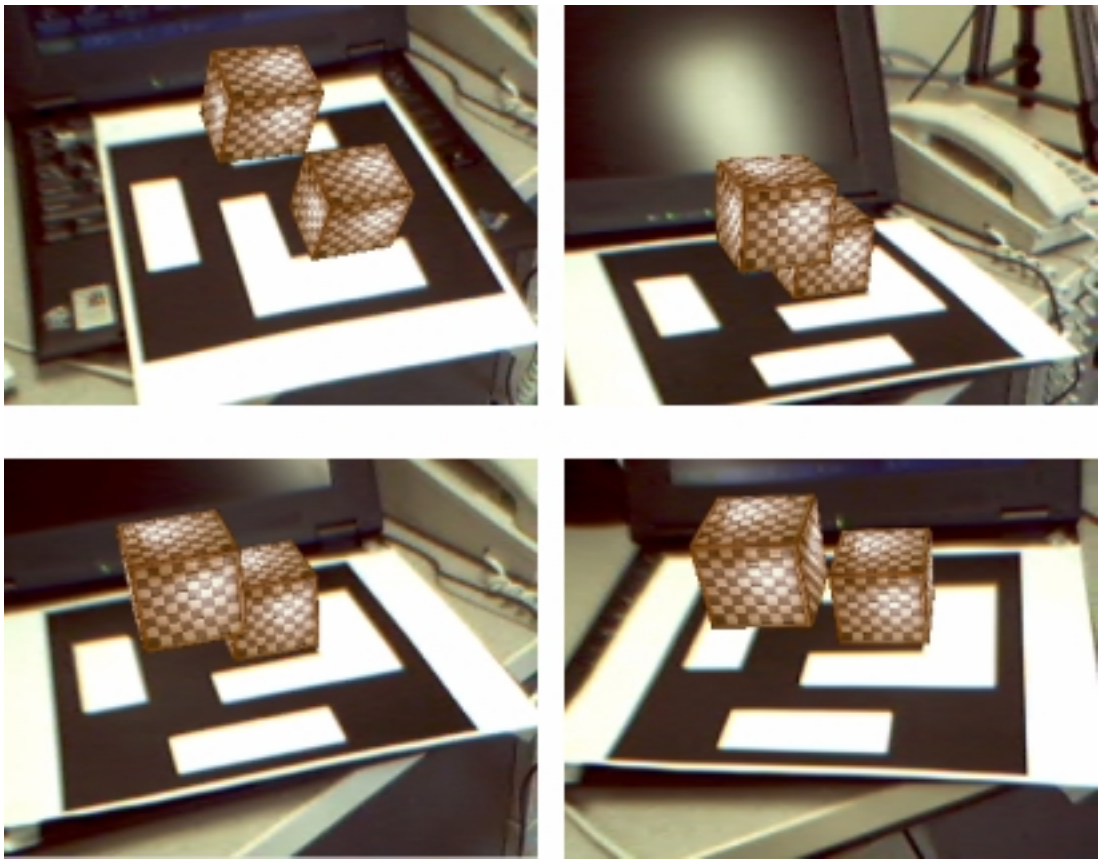


Figure 3.4 – Automatically calibrated 3D augmentation of two checkered cubes viewed from different viewpoints.

Chapter 4

Robust 2D Pattern Tracker Design

Chapter 3 described the basic augmentation system under the assumption that a set of valid correspondences between some planar pattern and image space could be found. This chapter proposes a tracking system that can be used to establish such correspondences. The two modes of the tracking system are first described, showing how a known pattern can be extracted from a frame of video and then tracked in subsequent frames. This is followed by a description of how the tracker facilitates the computation of accurate homographies. The chapter concludes by describing how the proposed system fulfills the five robustness criteria outlined in Chapter 1.

4.1 Defining a Pattern

The patterns used in our augmentation system consist of a thin white square border, inside of which is a thick black box consisting of any number of black or white corner features. For each pattern, a list of (x, y) coordinates representing the associated corner features is also generated. While any scale could be used for the X and Y axes of the pattern coordinate space, for convenience they should be in some metric units based on the size of the pattern as it would appear when printed on a sheet of paper. Additionally, the origin of the coordinate system should correspond to the centre of the pattern. As a result, the actual size of the pattern image in pixels is unrelated to the defined coordinate system. For our system, patterns are 64x64 pixels which provide enough space to define a sufficient number of features while only requiring a few kilobytes of run-time memory space. Figure 4.1 shows an example of a 64x64 pattern used by the system, with the axes of the metric coordinate space overlaid on top.

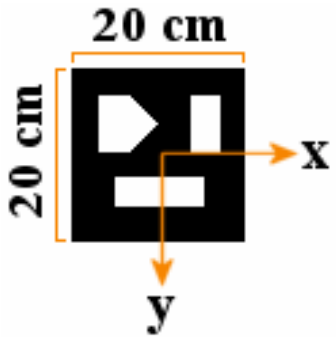


Figure 4.1 – Example of a 64x64 black and white pattern with corner features.

Since the pattern will ultimately be used to extract orientation information for augmentation purposes, it is important to assure that the arrangement of features within the black box of the pattern is asymmetric. In other words, if the pattern is rotated in 90 degree increments, its current orientation should be unambiguous. Since each pattern will be associated with a unique virtual object, the pattern should also be unambiguous in comparison to other patterns at each of the four orientations.

4.2 Tracking System Overview

Figure 4.2 shows the basic flow of the robust tracking system. The system starts in *Search Mode*, which involves using binary vision techniques over an entire frame of video in order to find valid patterns. This is followed by *Tracking Mode*, which allows fast and robust localized corner tracking of a pattern's individual features from frame to frame. Finally, if a valid pattern is being tracked, a homography is computed using the set of corner correspondences from pattern space into image space. The camera pose is then computed as discussed in Chapter 3 and the appropriate virtual 2D or 3D object that is associated with the tracked pattern is augmented onto the plane in the frame of video. Upon tracker failure, the system reverts back to *Search Mode* so that patterns can be detected again. The details of the system will be described in the following sections.

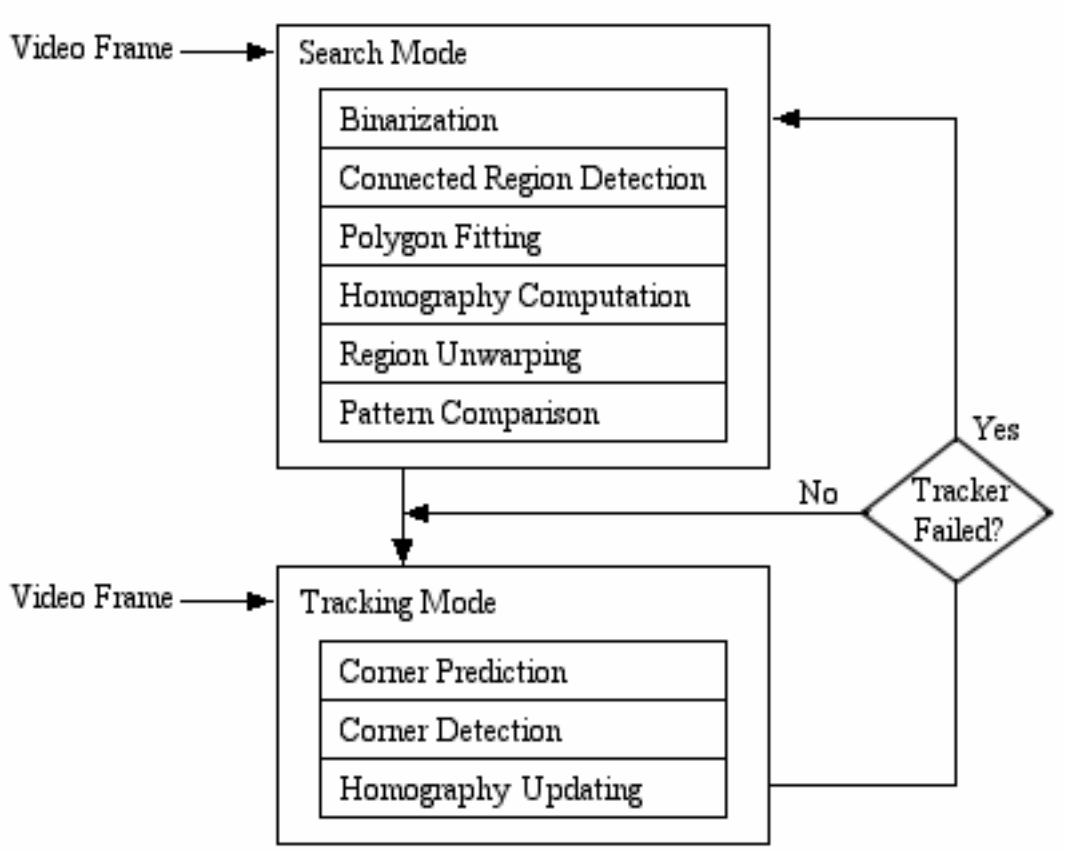


Figure 4.2 – Basic flow of the tracking system

4.3 Search Mode

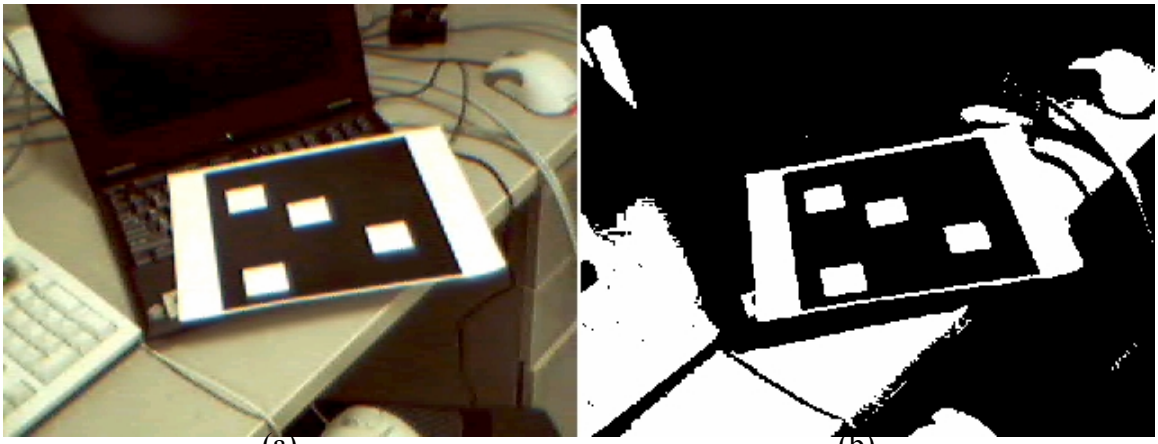
Before individual corner features can be tracked from frame to frame, the system must be able to decide which pattern, if any, is in the frame of video.

4.3.1 Candidate Region Detection

Many researchers have proposed using black and white planar patterns for augmented reality since features are defined using the highest contrast possible [REKI98, KATO00a]. Additionally, these high contrast features are extremely simple to extract from a frame of video using well-known binary image processing techniques [RUSS95]. Colour features, on the other hand, are prone to detection errors when attempting to achieve compatibility on a wide variety of low-cost cameras [MOLI01].

Binary Quantization

Therefore, in order to detect a potential pattern in a frame of video, we must first perform a binary quantization of the image. Figure 4.3a shows a pattern as it appears in a colour frame of video, and Figure 4.3b shows the same frame after binary quantization. Clearly, the black and white features have been preserved quite well, which is important for further extraction purposes.



(a) (b)

Figure 4.3 – (a) Original frame of video, (b) frame of video after binarization

Converting a colour image into a binary image is a trivial task. In our system, each frame of video is first scaled into a 320x240 image and then transformed into a grayscale image, such that each pixel has a value between 0 (black) and 255 (white). This grayscale image is then quantized into a binary image as follows:

$$P_B(x,y) = \begin{cases} 0, P_G(x,y) < T \\ 255, P_G(x,y) \geq T \end{cases}$$

where $P_B(x,y)$ is the (x,y) pixel intensity in the binary image, $P_G(x,y)$ is the (x,y) pixel intensity in the grayscale image, and T is the desired threshold value between 0 and 255.

While a fixed threshold value may work quite well, dynamic thresholding methods exist that allow the binary quantization system to adjust the T value based on changing scene conditions. The simplest approach involves generating a histogram from the grayscale image, and then analyzing it for the minimum and maximum intensity peaks. The

threshold *T* is then chosen to be the value between these two peaks. The assumption is that we would like the brightest pixels in the image to be categorized as white, and the darker pixels as black. Figure 4.4a shows an example of a grayscale scene, its associated histogram and *T* value, and the resulting binary image. The same scene under reduced lighting is shown in Figure 4.4b, showing the similarity in the final resulting binary image when using dynamic thresholding. Using a fixed threshold value, the binary image under reduced lighting is significantly different from the binary image under normal lighting.

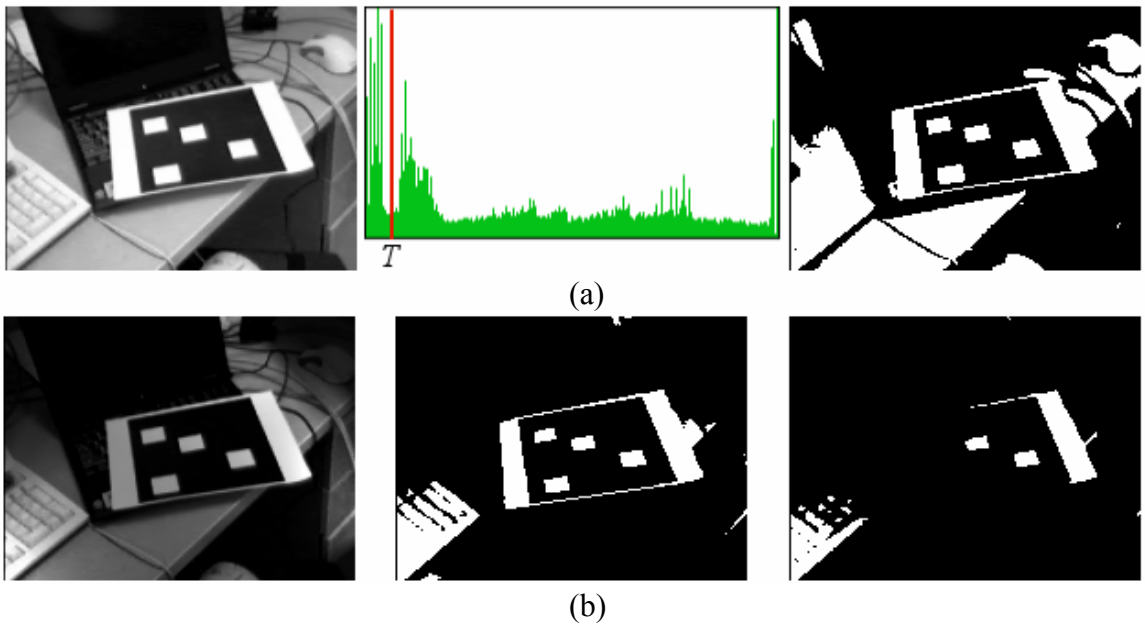


Figure 4.4 – Dynamic binary image quantization via histograms (a) under normal lighting using histograms, (b) under reduced lighting (left image is original, middle image is with dynamic threshold, right image is with fixed threshold).

Connected Region Analysis

Once a binary image has been generated it must be analyzed for potential pattern areas. Since we know the original pattern consists of a thin white border inside which is a thick black area, we must search for large connected regions of black pixels in the binary image. This can be accomplished using well-known flood-fill techniques from computer graphics as follows:

- Scan the entire image for black pixels, starting at the top-left of the image and scanning each pixel row-by-row.

- For each detected black pixel, mark it as scanned and set a bounding box around it.
- Recursively scan the eight neighbouring pixels for black, marking each one as being scanned and adjusting the bounding box accordingly.
- Once no more neighbouring black pixels can be found during the recursive step, label the computed bounding box as containing a region.
- Continue scanning the image pixels row by row.

In order to reduce further processing of invalid regions, we may apply the following simple heuristic checks:

1) If the bounding box around the region of connected pixels is smaller than some predetermined threshold, reject the region under the assumption that it may simply be a set of spurious black pixels. If the region actually is a valid pattern, the threshold states that the pattern may be too far from the viewer to provide any useful augmentations.
2) If the aspect ratio of the bounding box is below some predetermined threshold, reject the region under the assumption that it may be a long strip of spurious black pixels. If it turns out that it actually is a valid pattern, chances are that the plane is being viewed at an extremely sharp angle such that augmentations would eventually fail in later processing.

4.3.2 Convex Hull Fitting

Each candidate region thus far only consists of a set of pixels in an image and an associated bounding box. Ideally, we would like to fit a tight convex hull around the region that geometrically defines the region in the image.

We know that each correct pattern in the binary image will consist of a thick black border. Therefore, the best convex hull would consist of four points located at the corners of the candidate region.

Figure 4.5 shows some sample profiles of black image features in binary images. As can be seen, edges typically consist of approximately 50% black and 50% white pixels, while corners have more than 50% white pixels.

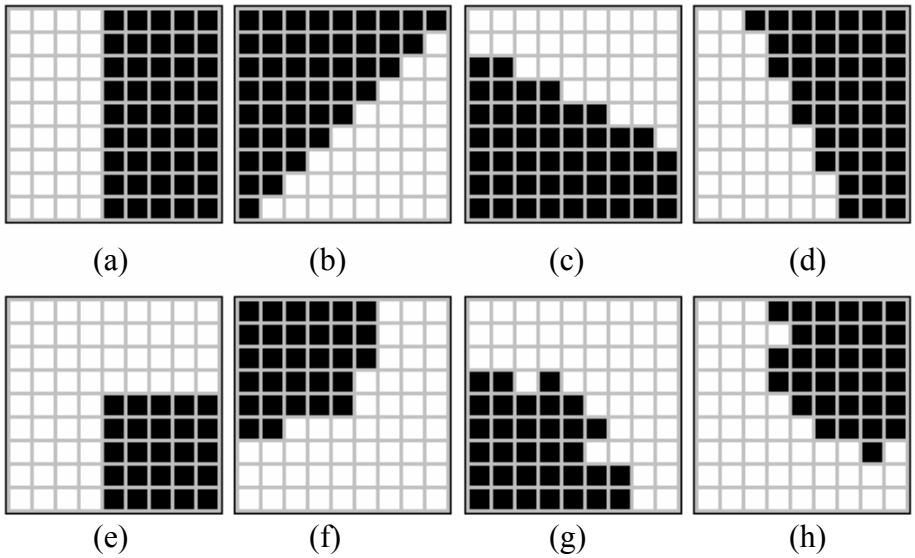


Figure 4.5 – Black image features in a binary image. (a)-(d) edges, (e)-(h) corners.

By slightly modifying the connected region algorithm in the previous section, a fast and simple heuristic approach can be used to detect the four strongest black corners for each candidate region. For each detected black pixel, we compute the following:

$$C(x,y) = N_W(x,y,S) / S^2$$

where $C(x,y)$ is a number between 0 and 1 representing the black corner strength of the (x,y) pixel. Here $N_W(x,y,S)$ returns the number of neighbouring white pixels for the (x,y) pixel in a local SxS search window. A typical window size for S is 9 for 320x240 images. The algorithm runs in $O(NS^2)$ time, where N is the number of pixels to be scanned.

By keeping track of the four highest scoring black corner pixels, we can quickly determine the vertices for the tightest fitting four-sided convex hull around any candidate region. After the four strongest corners for any candidate region have been determined, they can then be processed by a fast O(NlogN) convex hull computation routine in order to guarantee a clockwise or counter-clockwise vertex ordering [DEBE99].

4.3.3 Region Normalization

Now that the tightest fitting convex hull has been fitted to each candidate region, a final decision must be made as to whether the region is a valid pattern or not. Similar to the method outlined in [REKI98], the candidate region is unwarped such that it is front-facing, and then an image comparison with the set of known patterns is performed.

Fortunately, using our knowledge of homographies from Chapter 3, this proves to be a trivial task. Given the 4-sided convex hull in image space that was discussed in the previous section, a set of vertex correspondences between the convex hull and the known 2D coordinates of the original patterns can be made. These correspondences can then be used to solve for H, which represents a 2D-to-2D mapping from the original pattern plane into the detected planar region in image space. H can then be used to generate a normalized (unwarped) image as follows:

```
For each pixel location p = (x,y) in the original pattern image I_P
      Compute p' = H p
      Get the pixel intensity at p' in the binary video frame I_B
      Store this pixel intensity at p in the normalized image I_N
```

This results in a new image I_N of the same dimensions as the pattern image I_P, but consisting of pixels from the binary video frame I_B such that the detected region defined by the convex hull is now front facing (perspective distortion and orientation are removed).

Before the normalized image can be matched with the set of known patterns, orientation must be taken into account since the order of the convex-hull vertices do not necessarily correspond with the original pattern vertices. For example, Figure 4.6a shows a captured video image, and Figure 4.6b shows the normalized region. Notice that the vertices v_0', v_1', v_2', and v_3' as computed by the convex hull do not map directly to the correct vertices of the original pattern in Figure 4.6c. The correct mapping should actually be $v_0' \rightarrow v_1$, $v_1' \rightarrow v_2$, $v_2' \rightarrow v_3$, and $v_3' \rightarrow v_0$. To guarantee that the proper orientation is

found, four different homography computation and normalization procedures must be computed, one for each mapping.

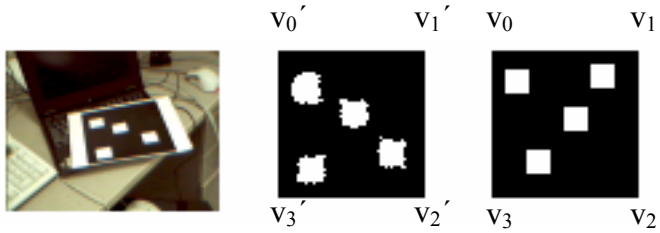


Figure 4.6 – Region orientation and vertex ordering of the convex hull. (a) Captured video frame, (b) Normalized region and vertex ordering, (c) Original pattern vertex ordering.

4.3.4 Pattern Identification

Now that a candidate pattern region has been identified in a video frame and has been normalized to be front facing, a final determination can be made as to whether this region contains a valid pattern or not.

Given a normalized region image I_N as computed in the previous section, a correlation score can be computed with each original known pattern I_P as follows:

$$score = \sum_{i=0}^{w \times h} I_P(i) \oplus I_N(i)$$

where i represents a pixel location, and w and h represent the width and height of the pattern images respectively (in our case $w, h = 64$). In other words, we determine a score based on summing the absolute difference between pixel intensity values. Since both images are binary (black or white), we can use the exclusive-or operator to compute the absolute difference. In this case, a lower score represents a better match, and a zero score means the images are identical. A threshold is used to decide on whether this correlation operation has produced a match or whether the region should be rejected.

The end result of the correlation operation is an association between each original candidate region in screen space and the best matching known pattern in pattern space (if any).

4.4 Tracking Mode

The previous section outlined a fast approach to self-identifying known planar patterns in a frame of video. Existing methods proposed in [REKI98] and [KATO00a] consider this to be sufficient in order to begin augmenting virtual objects onto the pattern. In other words, camera parameters can be extracted from the homography that was computed for region normalization purposes, and 3D objects can now be augmented (as discussed in Chapter 3). The advantage of this approach is the simplicity of the system. However, as soon as any portion of the planar region is occluded (by the extents of the screen or by a foreground object, for example), the detection process completely fails. For interactive augmented reality this is completely unacceptable, since we will eventually want to manipulate virtual objects using hand gestures or pointing devices.

This section proposes an efficient corner tracking algorithm which allows continued pattern tracking in the presence of significant occlusion. The basic idea involves tracking the known corner features for a detected pattern from frame to frame, and robustly computing the homography for the planar pattern based on these interior corner features, instead of simply relying on the four outside corners of the black pattern border. Additionally, since local features are tracked from frame-to-frame, an increase in performance can be expected.

4.4.1 Coarse Corner Prediction

The first stage of *Tracking Mode*, as depicted in Figure 4.2, consists of corner location prediction using the homography computed in the previous frame. If the system was previously in *Search Mode*, this homography is the one used to normalize the planar region.

Using this previous homography H, all known corner positions $\mathbf{x_i}$ for the detected pattern are transformed into the current image space. Assuming temporal coherency, first-order prediction is used to translate the transformed position by an amount equal to any motion that corner experienced between the previous two frames.

In other words, a predicted location $\hat{\mathbf{x}}_\mathbf{i}$ is determined for each corner as follows

$$\hat{\mathbf{x}}_\mathbf{i} = H\mathbf{x}_\mathbf{i} + v_i \Delta t$$

where v_i is the current velocity of corner i, and Δt is the time elapsed since the last frame.

Each of these predicted corner locations are inaccurate due to small errors in the homography as well as due to unexpected camera or pattern motion since the last frame. Figure 4.7 shows an example of a set of predicted corners using the previous homography where the actual corners in the current frame have moved slightly.

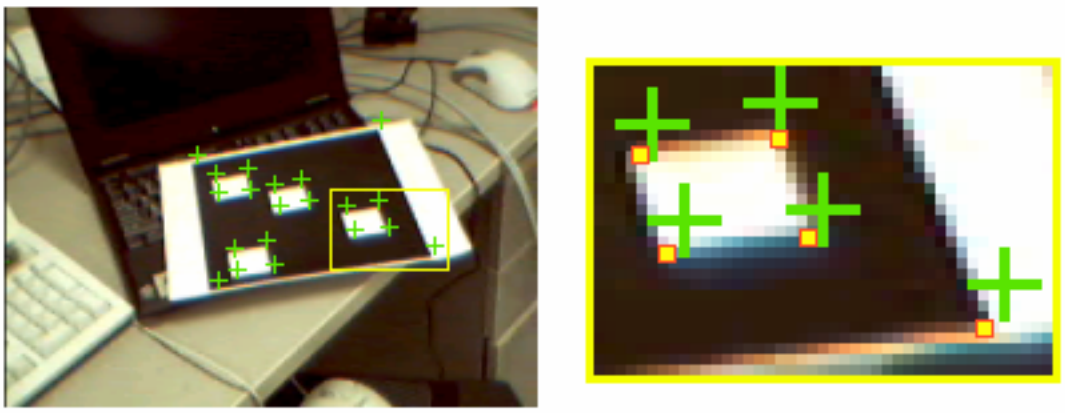


Figure 4.7 – Corner prediction. The left image shows the predicted corners for a scene using the previous frame's homography. The right image shows a zoomed view of the yellow area in the left frame. The green crosshairs represent the predicted locations, and the yellow squares represent the actual corner locations.

4.4.2 Subpixel Corner Detection

In order to find the actual corner positions for each of the known pattern corners $\mathbf{x}_i$ in the current frame, a local search window *W* is placed around each predicted corner location $\hat{\mathbf{x}}_i$, and the actual corner $\mathbf{x}_i'$ must be in this window. Using a reliable corner finding algorithm with subpixel precision (see Appendix A), the strongest corner $\mathbf{x}_i'$ is then extracted from this search window and a correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$ is made. If no corner is detected, the corner feature ***i*** has its tracking status marked as having failed for this frame. This could occur due to the corner being off the edge of the screen or temporarily occluded.

Note that extracting the strongest corner may not always be the best heuristic since ambiguities may occur when the search window overlaps more than one valid corner, as depicted in Figure 4.8. Another heuristic is to choose the corner in the search window that is closest to the predicted location. However, this still does not always guarantee that the correct corner will be chosen. A more sophisticated solution was presented in [BRAN99], whereby neighbouring corner positions and edge properties from the original pattern are taken into account to determine the best corner location. Another approach involves computing a cross-correlation between a corner's current and previous frame search windows, but this is computationally too expensive for real-time performance.

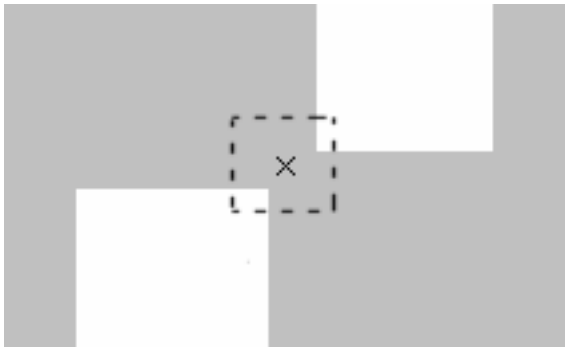


Figure 4.8 – Multiple potential corners in a single search window.

The search window size defines how much pixel movement the tracker can tolerate from frame to frame, assuming smooth motion. A large search window allows for more

movement, but increases the chances of multiple corners being found in the search box. Conversely, a small search window size reduces the chance of multiple corners, but increases the chance of tracker failure due to rapid motion.

In order to account for these discrepancies, we use a dynamic search window in the pattern which changes size in proportion to the area occupied by all the corner features. In other words, if the pattern is determined to be far from the viewer (small pattern area), reduce the search box size under the assumption that 1) significant motion of the pattern has a smaller effect in screen space, and 2) corners in the pattern will be closer together. If the pattern is found to be close to the camera (large pattern area), increase the search box size since the chance of multiple corner detections is lower and pattern motion has more effect in screen space.

In our experiments we found that when the pattern covers approximately 25% of the pixel area in our 320x240 images, a 20x20 search window allows for a sufficient amount of movement while still allowing corners to be determined unambiguously.
Thus the following simple scaling approach can be used to compute the dynamic search window dimensions

$$W_{width} = \frac{20}{160} B_{width} \qquad W_{height} = \frac{20}{120} B_{height}$$

where B represents the bounding box, in screen space, around the predicted corner locations.

The next section discusses how the set of accurate corner correspondences that were found are used to compute a robust homography for subsequent tracking and augmentation purposes.

4.4.3 Homography Updating

Using the well-known RANSAC approach for robust estimation [FISC81, SIMO00], a new homography for the set of accurate corner correspondences $\{\mathbf{x} \leftrightarrow \mathbf{x}'\}$ is determined as follows:

- Randomly sample four non-collinear $\mathbf{x} \leftrightarrow \mathbf{x}'$ correspondences.
- Compute H from the random sample.
- For all correspondences, compute the distance between $\mathbf{x}'$ and $H\mathbf{x}$.
- Count the number of pairs for which the distance is below some threshold. A value between 2.0 and 4.0 works well for our system. These correspondences are considered our inliers, and the rest are labelled as outliers.
- Store the H which has the highest number of inliers I_H.
- Refit the homography H using these inliers.

The motivation behind random sampling is to remove inaccurate or mismatched corner locations from the homography computation. This allows the homography to be robust to partially occluded features, which is important for subsequent corner predictions and stable augmentations.

The number of random samples is capped at the maximum number of known corners I_{max} for the detected pattern. Most of our patterns consist of four to eight white rectangles, resulting in 16 to 32 corners. In order to reduce the number of samples further, random sampling stops if the highest number of inliers I_H is above the following threshold

$$I_H \geq T_Q I_{max}$$

where T_Q defines a quality measure between 0 (low) and 1 (high). A value above 0.75 provides relatively stable homographies.

The best homography computed after random sampling is then used as the corner predictor in the next frame. Note that random sampling can fail under the following instances:

- There are less than 4 non-collinear $\mathbf{x} \leftrightarrow \mathbf{x}'$ correspondences.
- The best computed H fails to meet our T_Q criterion
- I_H falls below 4, which is the minimum number of correspondences required for computing H in the next frame.

In such cases, the tracking system reports tracker failure and reverts back to *Search Mode*.

The basic idea behind updating the homography via random sampling is to increase the robustness of the pattern tracker. With multiple corners being tracked simultaneously, occlusions of a subset of the feature points should have little or no effect on the corner prediction in the next frame. For example, if a detected pattern consisting of 24 corners is being tracked, the homography should still be able to predict corner positions for the next frame even if approximately 16 corners are currently occluded by a user's hand.

The assumption is that the other 8 corners interspersed around the pattern area are sufficient to accurately determine the pattern's current orientation. In fact, 4 unoccluded, non-collinear corners are all that is necessary. The predicted locations for the occluded corners will still be searched in upcoming frames so as soon as the occlusion stops (i.e. the user's hand is removed from the pattern) all 24 corners will be detected again.

4.5 Augmentation in 2D and 3D

As discussed in Chapter 3, once the homography H has been computed it can be used to augment any virtual 2D or 3D object onto the projection of the pattern in the video frame.

Figure 4.9 shows the outline of our video see-through augmentation system. A frame is captured from the video camera, and the corner tracking system is used to robustly compute a homography. A perspective projection matrix is then extracted from the homography and then passed to the OpenGL graphics library. OpenGL is used to render

the virtual 3D objects onto the original video frame. The augmented frame of video is then presented to the user on either a head-mounted display or on a standard computer monitor.

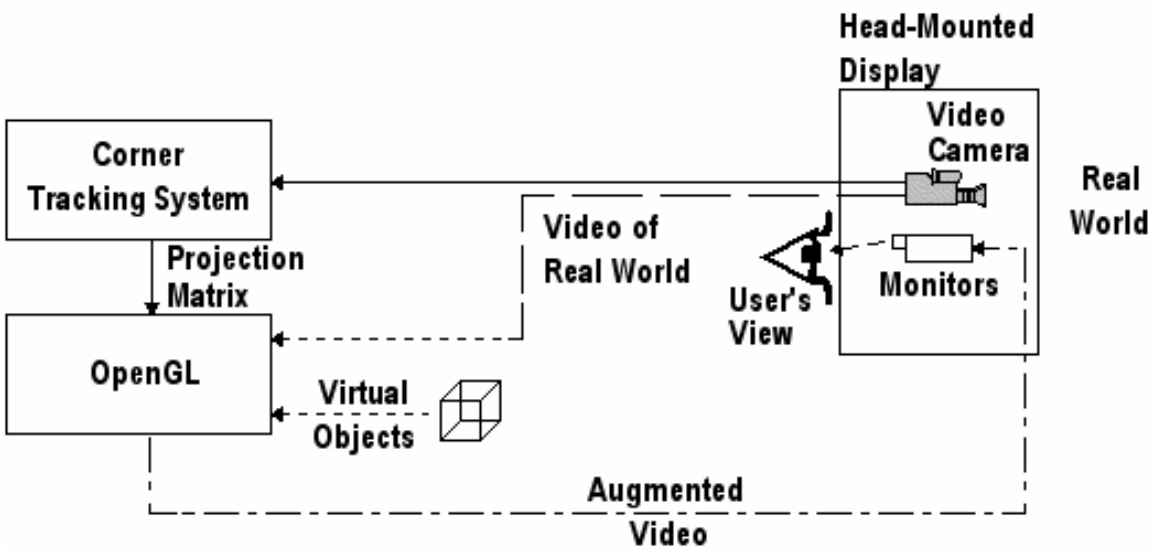


Figure 4.9 – Augmentation system outline

4.5.1 Weighted Autocalibrations

The camera parameter estimation formulae in Section 3.3 are directly dependent on the quality of the homography *H*. It is possible that the computed focal lengths change significantly from frame to frame due to errors in the computation of *H*. As a result, virtual 3D objects have the potential to jitter or shake, resulting in unrealistic augmentations. In order to reduce this effect, it would be convenient if the estimated camera parameters were stabilized during the tracking phase based on the quality of the computed homography.

For example, when a planar pattern is relatively front-facing, the computed *H* accurately defines the planar region. At extremely sharp angles, however, the computed *H* may contain significant error since corresponding feature points are extremely close together or almost collinear.

Since a single estimate of the camera focal length is not necessarily accurate we average these estimates over time as follows:

- Use the autocalibration formulae from Chapter 3 to compute the intrinsic camera parameters (the focal lengths in the two axes) for each frame in the live video.
- Determine a weight for the computed focal lengths based on the current quality of the homography.
- Insert the focal lengths into a list that is sorted by weight in descending order, where the list contains a maximum of K entries.
- Use the entries of the sorted list to compute a weighted average of the focal lengths.
- Use these weighted average to compute the extrinsic camera parameters using the autocalibration formulae of Chapter 3.

The weighting factor of the focal lengths is based on the reprojection quality of the homography H. Given each original pattern corner $\mathbf{x_i}$, the distance between $H\mathbf{x_i}$ and the predicted location $\mathbf{x_i}'$ is determined. The average of these distances gives us the homography's average reprojection error, RE_{avg}, represented mathematically as

$$RE_{avg} = \frac{\sum_{i=1}^{N} \left\| \mathbf{x_i}' - H\mathbf{x_i} \right\|}{N}$$

where N is the total number of corner correspondences.

Since a good homography should give us low values for RE_{avg}, the weight assigned to each autocalibrated focal length is $\frac{1}{RE_{avg}}$.

By averaging the focal lengths we stabilize the intrinsic parameters which would otherwise vary slightly with each image. This also has the effect of stabilizing the 3D augmentation.

4.6 System Summary and Discussion

In this chapter we presented an algorithm to detect and track planar patterns in real-time, with the design largely driven by the five main robustness criteria described in Chapter 1.

Speed: The binary image processing techniques used in *Search Mode* can be performed on today's consumer-level PCs at real-time rates. However, since the system will normally be in *Tracking Mode*, this is where performance is critical. While a corner finding algorithm performed on an entire image at every frame is not possible at interactive frame rates, the use of localized search windows allows for real-time performance. The determining factor is the number of corners that need to be tracked for any given pattern.

Self-identification: The proposed system uses binary image processing and matching techniques to uniquely identify planar patterns in a frame of video. Therefore different patterns can be viewed in the augmented environment, and the system can determine the correct virtual object that should be augmented onto this pattern.

Scale and orientation invariance: Once a pattern has been detected in the frame of video, individual corner features are tracked using dynamic-sized localized search windows. Corners are naturally invariant to scale, since corners are based on intensity changes around a pixel. Therefore, as long as the edges defining the corner are still visible, the corner will remain detectable. Additionally, due to the use of a dynamic local search window size, the tracker is also tolerant of a significant amount of change in the orientation of the pattern.

Robustness to occlusion: The use of random sampling when computing a new homography in *Tracking Mode* allows for significant occlusion of the pattern, which is important for when only part of the pattern is visible. As long as four non-collinear corners are still being successfully tracked, augmentation of virtual objects will be possible.

Robustness to lighting: *Search Mode* is robust to changes in lighting due to the use of a dynamic threshold value for binary quantization. In *Tracking Mode*, corner features are naturally robust to lighting changes since the corner finder looks for edge intensity gradients in two directions to determine corner strength (Appendix A).

Chapter 5

Analysis

This chapter describes the performance and stability of our tracking and augmentation system. The primary goal of our work is to develop a practical system that can be used for interesting augmented reality applications. For this reason we have developed an implementation that allows us to test the system with respect to the five robustness criteria described in Chapter 1.

5.1 Performance

One of the major design goals of our augmented reality system is real-time performance on standard PCs using off-the-shelf USB camera hardware. The current implementation of our tracking system uses OpenGL to augment simple 2D textures and 3D objects onto the planar patterns at 20Hz on an Intel Pentium 3 800MHz PC equipped with an ATI Rage 128 Video card and an Intel CS110 USB camera.

In order to obtain an accurate estimate of the *Search Mode* time, we modified the implementation to always remain in *Search Mode* and to use the computed homography to perform augmentation.

The current breakdown of average processing time per frame when viewing a static planar pattern consisting of 24 corners and a 2D augmentation is as follows:

Search Mode: 29.1ms
Tracking Mode: 10.7ms
Augmentation Time: 2.1ms

Clearly, the global search method is significantly slower than tracking 24 localized corner features. However, the system is usually in *Tracking Mode* where performance is directly related to the number of corners being tracked; the more corners we have, the longer it takes for the tracker to operate. Figure 5.1 shows tracking times for a set of patterns with varying corner counts. As can be seen, processing time increases approximately linearly with respect to the number of corners that need to be tracked.

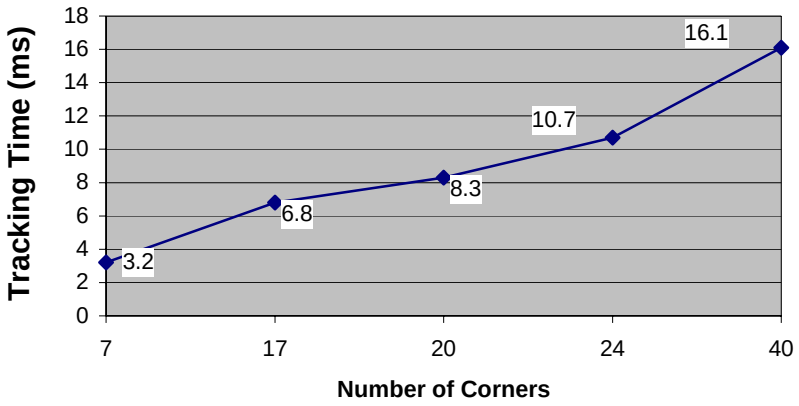


Figure 5.1 – Corner Tracking Performance

5.2 Self-Identification

An association between a pattern in a frame of video and a virtual 2D or 3D object can be made as long as all patterns are defined unambiguously (as discussed in Section 4.1). Therefore, the system is capable of recognizing different patterns and unique virtual objects can be augmented onto each one. Our current implementation uses either randomly placed white boxes or manually placed white polygons on a black square background, with the positions of the boxes or shapes stored in a data file. As a result, there is no accurate formula to estimate the number of uniquely definable patterns. A theoretical estimate can be made, however, by setting some simple constraints on the arrangements of the white boxes.

Let us assume that each 64x64 pattern consists of a black border that is 8 pixels wide. Therefore, each pattern provides a 48x48 area inside which we can define our white boxes. Assume that these boxes are fixed at 12x12 pixels, and that they must be placed

on 12 pixel boundaries in the 48x48 area. This provides a 4x4 grid of available slots into which white boxes can be placed. In order to guarantee that the patterns are unambiguous at the four possible pattern orientations, we can further assume that the top-left 12x12 box is black, the top-right is white, the bottom-right is white, and the bottom-left is black. Therefore we have 12 available boxes that can be coloured either white or black to define a unique pattern. Figure 5.2 shows the constrained pattern arrangement, allowing for up to 2^{12} uniquely identifiable patterns.

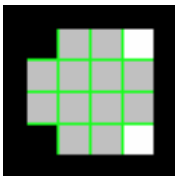


Figure 5.2 – Constrained pattern arrangement allowing up to 2^{12} unique patterns (grey boxes can be either black or white)

Note that the number of available corner features that can be tracked in these 2^{12} patterns varies based on the arrangement of the white boxes. Therefore the patterns are not guaranteed to perform equally with respect to the other robustness criteria.

5.3 Scale Invariance

The major advantage in using corners for tracking is that corners are invariant to scale. The corner tracker can thus continue to track patterns at a large range of distances from the camera. The use of a dynamic search window size for corners facilitates this process, since the search windows become smaller as the area occupied by the corner features in a particular pattern begins to shrink.

Table 5.1 shows the range of scale allowed for the set of test patterns shown in Figure 5.3. Patterns differ in the number and/or arrangement of corner features. Scale is measured based on the percentage of screen area covered by the bounding box around the tracked corners. Each pattern was first recognized in *Search Mode* and then placed such that it covered approximately 50% of the view in a front-facing position. The pattern was then slowly moved away from the camera until tracking failure, after which the minimum

percentage of occupied screen space during tracking was computed. The process was repeated again for maximum scale by slowly moving the pattern closer to the camera instead of away from it.

A B C D E F G

Figure 5.3 – Test patterns for tracking experiments

Table 5.1 – Distance (scale) tracking tolerance for test patterns

Pattern	# of Corners	Min % of Screen	Max % of Screen
Pattern A	7	1.4	102
Pattern B	17	1.3	190
Pattern C	20	1.6	257
Pattern D	20	1.6	236
Pattern E	22	1.3	275
Pattern F	24	1.2	218
Pattern G	40	1.5	370

As Table 5.1 shows, all the patterns allowed tracking to continue when the pattern occupied less than 2% of the screen area. At such small scales, however, any significant movement of the pattern causes tracking failure since the corner search windows are extremely small. Some experiments involving pattern movement at large distances showed that patterns occupying at least 10% still allow significant pattern movement. The number of corners in a pattern does play a role when a pattern covers a large area of the screen, as can be seen in Table 5.1. Pattern A, with only 7 corners, only allows the pattern to cover 102% of the screen area, while Pattern G allows the pattern to cover 370% of the screen since more corners continue to be visible at this scale. Therefore, increasing the number of corners does not appear to provide any significant improvement when viewing patterns far from the camera. However, when viewing a pattern close to the camera, extra corner features that are close together are beneficial since the majority of the pattern is outside the visible areas of the captured image.

Currently, patterns must be visible in at least 25% of the view, with no occlusions, in order for *Search Mode* to lock onto the pattern. This can be adjusted based on size and aspect ratio thresholds. Figure 5.4 shows the range of distances in which a front-facing pattern with 24 corners can be tracked successfully. Note the dynamically changing size of the search windows.

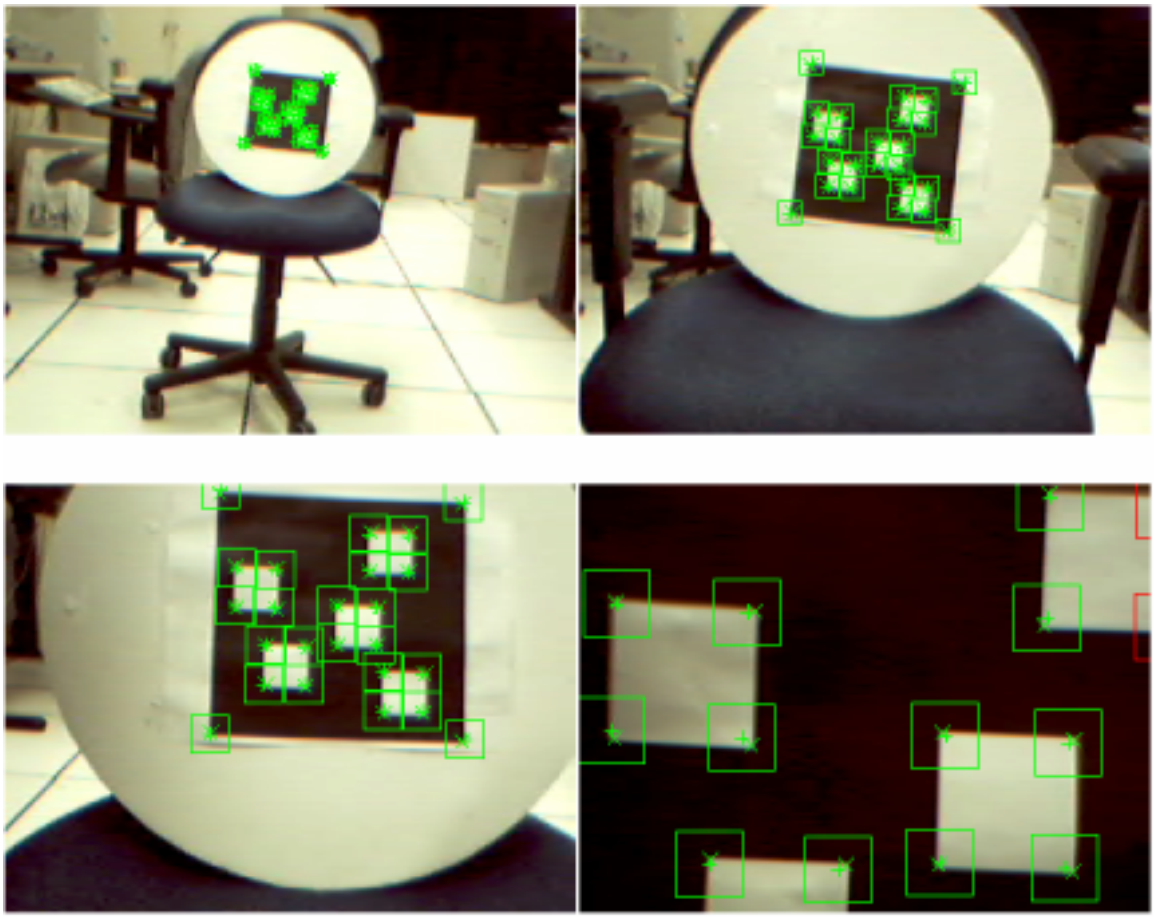
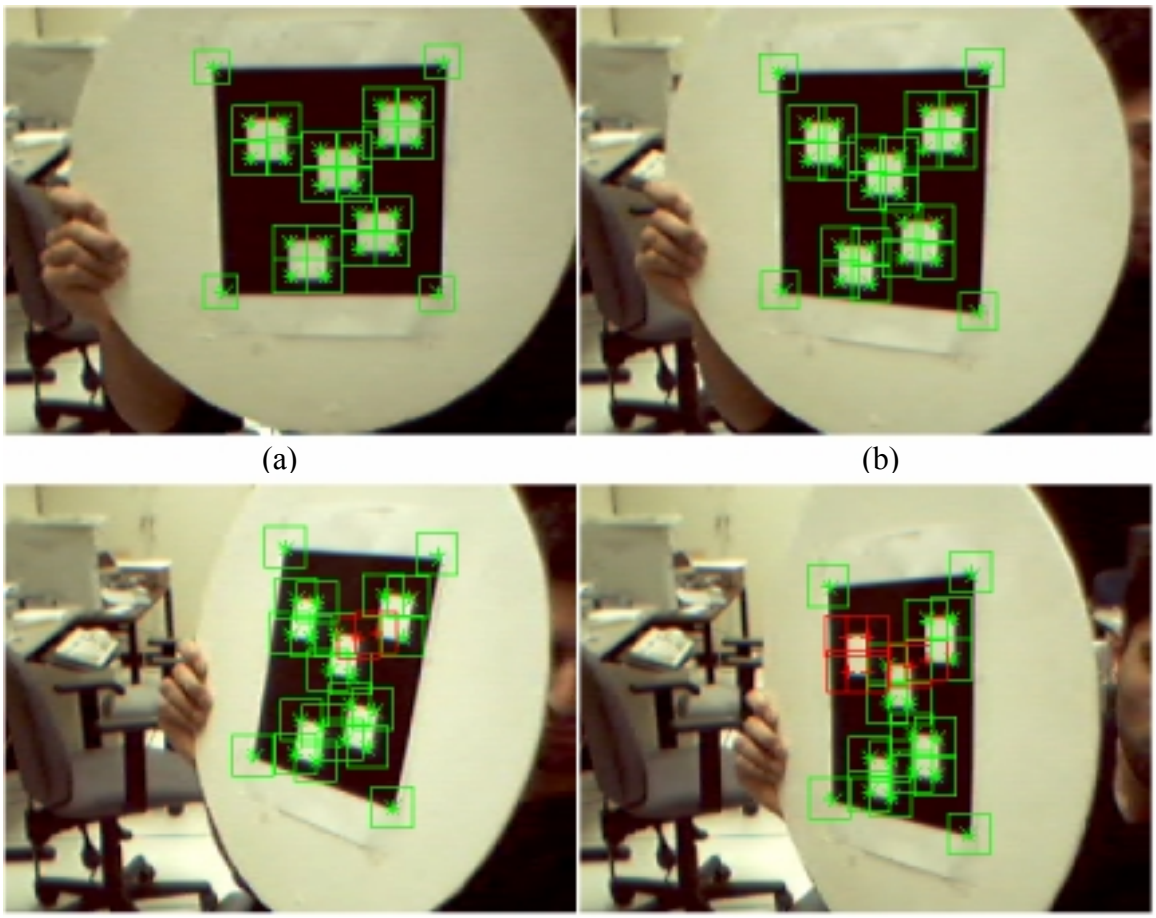


Figure 5.4 – Allowable range of distances for a 20-corner pattern

5.4 Orientation Robustness

Due to perspective distortion, a square on the original pattern does not necessarily remain square when viewed at a sharp angle and projected into image space. Thus, corners will change their shape which affects the corner finder's detection ability. However, this is not a problem since the corner finder our system uses relies on finding intensity changes in two directions (Appendix A), which are still evident even at severe orientations.



(a) (b)

(c) (d)

Figure 5.5 – Tracking under severe rotation (green boxes are successfully tracked corners, red boxes are corners that have failed to be detected unambiguously)

The major source of orientation problems is the corner search window size. It plays a key role since corner features on a planar region begin to converge onto other corners in screen space as pattern rotation increases. For example, Figure 5.5 shows a 24-corner pattern undergoing a gradual rotation of 90 degrees. As can be seen, the corner detection begins to deteriorate when corners enter search boxes for other corners. As expected, the average reprojection error of the homography also begins to increase as more corners begin to converge. However, as Figure 5.5 shows, the tracking continues up to almost 75-80 degrees, which is quite robust for many types of augmentations.

Figure 5.6 plots the reprojection errors of the various patterns as they undergo a rotation of 90 degrees, in approximately 15 degree increments, from the front-facing position.

The results show that increasing the number of corners negatively impacts the ability of the pattern to be rotated. This is due to the fact that under severe rotation, a large number of the search windows begin to overlap other valid corner features, leading to ambiguous corner detections as discussed in Section 4.4.2.

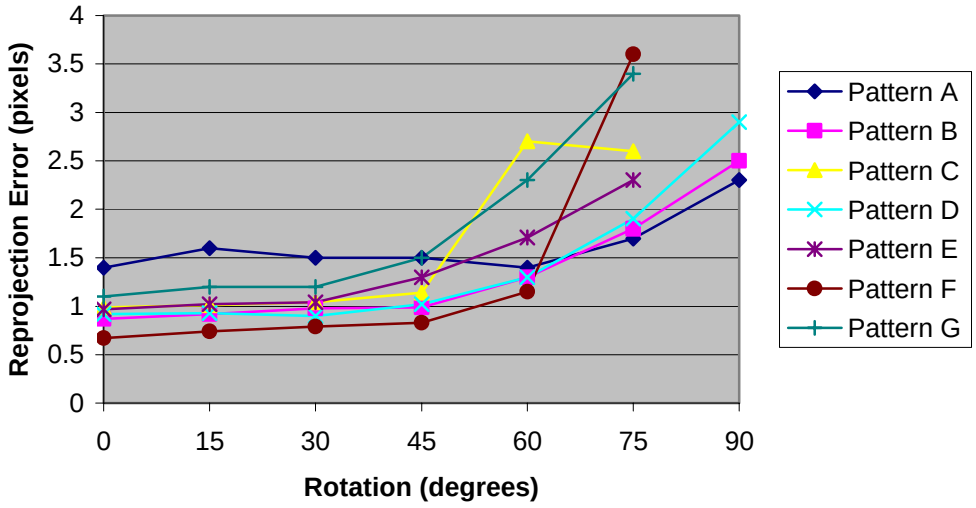


Figure 5.6 – Tracker rotation tolerance

5.5 Occlusion Robustness

One of the main advantages of our tracking approach is the ability to handle significant amounts of occlusion. Figure 5.7 shows a 17-corner pattern experiencing approximately 30% occlusion. The tracker can still detect enough corners so that a virtual 2D image can be correctly augmented onto the plane.

Figure 5.7 – Example of pattern occlusion

Even under motion, the prediction scheme allows corners to be correctly recovered after temporary occlusions. Figure 5.8a shows a pattern with all of its 20 corners being tracked successfully (indicated by green corner search boxes). Figure 5.8b then shows a hand occluding two of the corners, with red boxes denoting the predicted locations. While occluded, the pattern is rotated slightly such that one of the occluded corners becomes visible again. The non-occluded corner is then recovered, as depicted in Figure 5.8c. Note that the other corner is still being occluded, but the predicted location has changed after rotation. After removing the hand from the scene, the predicted location for the remaining corner allows it to be recovered, as depicted in Figure 5.8d.

Our results show that occlusion robustness is directly related to the number of corner features available for tracking; the more corners a pattern has, the more tolerant it is of partial occlusions. In our experiments, Pattern G (with 40 corners) allows for the most occlusion. This was also evident in our scale experiments, where it was observed that a larger number of corners allowed for patterns to be viewed at close range (when many corners were occluded by the edges of the visible screen area). Another interesting observation is the gradual decrease in the stability of the computed homographies as more corners become occluded in any given pattern. Predicted locations for corners begin to jitter significantly when the corner counts fall below 10 or so, depending on the collinearity of the remaining unoccluded corner features.

Note that only *Tracking Mode* provides any robustness to occlusion. *Search Mode* always requires the outside black border of the pattern to be completely visible.

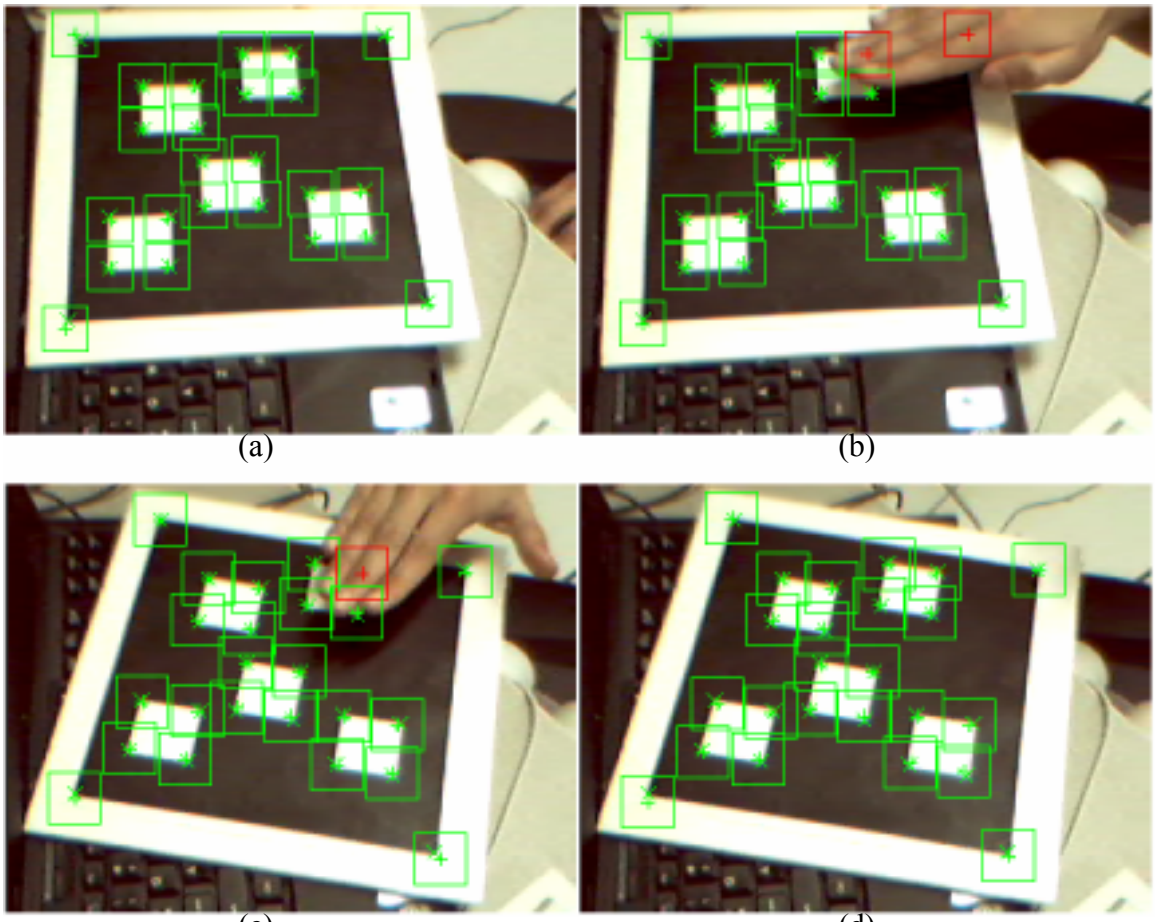


Figure 5.8 – Corner recovery from occlusion after pattern motion

5.6 Lighting Robustness

Another favourable property of the corner feature for tracking purposes is its robustness to lighting. Since intensity changes in two directions form the basis of corner finding, no special considerations need to be taken into account from the tracker's perspective in order to handle significant lighting changes or shadows. Therefore, as long as the black and white features on the planar pattern are somewhat distinguishable, the corner tracker continues to compute an accurate homography, even under dramatic changes in lighting (as shown in Figure 5.9).

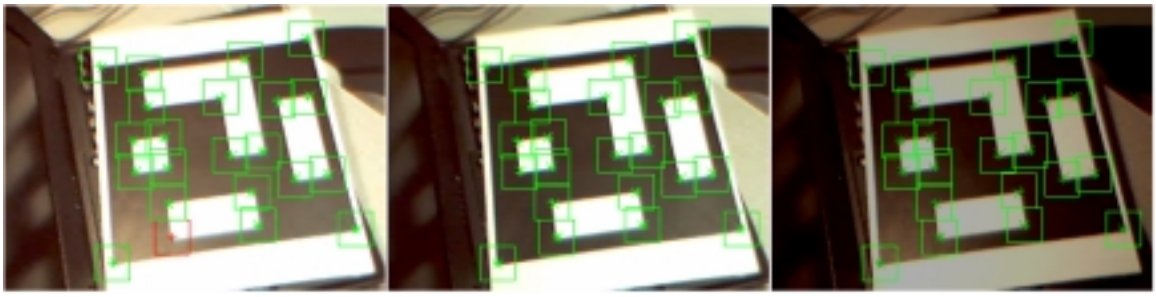


Figure 5.9 – Continuous corner tracking under lighting changes

5.7 Registration Stability

The stability and accuracy of the homography is directly related to the stability and accuracy of the augmented objects. In other words, if the homography cannot accurately recreate the orientation of the pattern in screen space, then the pose computations will be similarly affected. In order to test the stability of the system we use the average reprojection error RE_{avg} described in Section 4.5.1.

Our current implementation exhibits an average reprojection error between 0.67 and 1.4 pixels with various front-facing patterns (all corners being tracked successfully), which is quite good for our captured 320x240 images and a standard USB camera. The reprojection error begins to increase gradually as the orientation increases from the front-facing position. Occlusion of some corner features also increases the reprojection error, while scale does not seem to have any major effect.

5.7.1 2D Registration

As discussed in Chapter 2, there are two types of registration errors: static errors are alignment errors and jitter that occur when viewing a virtual object in an unchanging scene, while dynamic errors are those which occur when a scene is in motion.

In our system, static errors for 2D registration are due to inaccuracies in the homography, which are a direct result of inaccuracies in the detected positions of the corner features. In order to test the amount of jitter, a number of different patterns were viewed with the

camera in a fixed position. For each detected corner position in a pattern, the amount of pixel movement was recorded from frame to frame and the average pixel drift was computed.

The system currently exhibits an average pixel drift of 0.16 pixels, regardless of the pattern, which is quite accurate for our standard USB camera hardware. Neither scale nor orientation has any major effect on the average pixel drift.

Since a video see-through technology is being used in our system, the dynamic error during 2D registration is the delay between the time a frame is captured from the camera and when the final augmented scene is presented to the user. Currently the system performs at approximately 20 Hz while tracking, which leads to a delay of 50ms. For most applications this is acceptable, but, as discussed in Chapter 2, anything above this delay would reduce the immersiveness of the augmented environment.

5.7.2 3D Registration

The static and dynamic registration errors that were described for 2D registration also occur when augmenting 3D objects. However, 3D registration exhibits additional dynamic registration errors.

Since the projection matrix is defined relative to the stored pattern, errors in the projection matrix become more noticeable the farther the augmented object is from the 2D pattern plane. As a result, when a pattern is rotated in the scene, the homography is constantly changing and thus the computed Z axis vector for the pattern changes. Therefore, augmented objects occasionally experience jitter in vertices that are far from the pattern plane when the camera or pattern is in motion. When augmented objects have a relatively small change in depth, there is little visual impact of small errors in the projection matrix.

A related dynamic error associated with 3D registration is the variance in focal lengths during the camera autocalibration. At various orientations, inaccuracies in the homography cause the computed focal lengths to change significantly from the last frame, causing virtual objects to stretch or jitter during camera or pattern motion. The weighted autocalibration scheme helps to resolve such sudden changes. Figure 5.10 shows the variance between the current and weighted focal lengths for a 30 second sequence in which a pattern is rotated left and right by 60 degrees from the front-facing position. As can be seen, the weighted focal lengths help reduce the sudden focal length changes that occur during pattern motion.

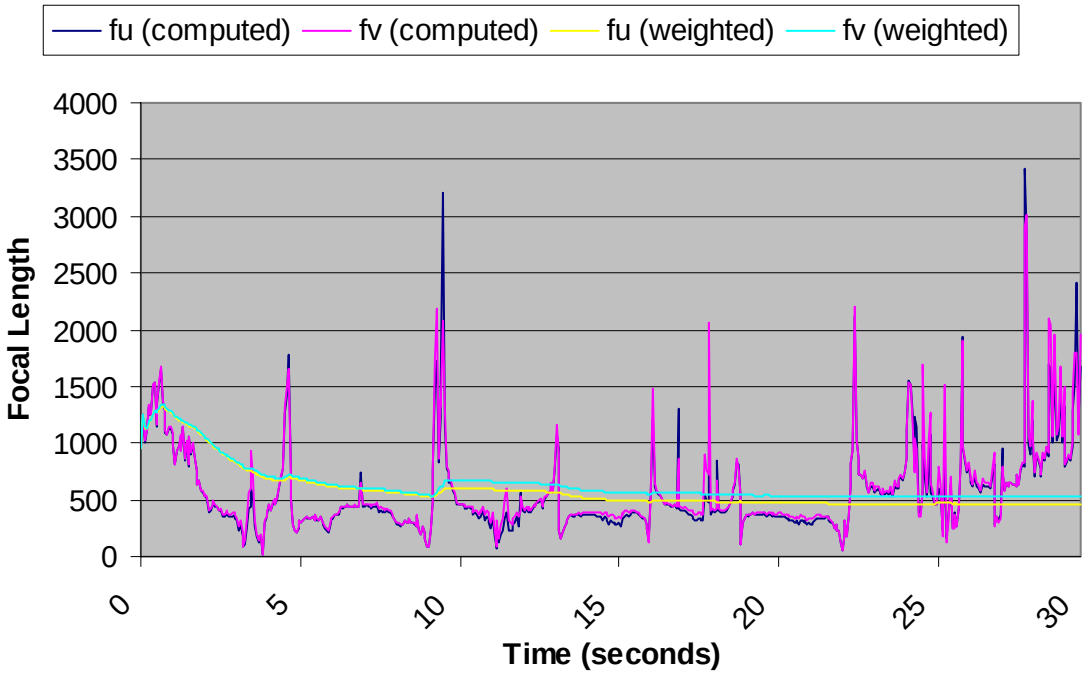


Figure 5.10 – Variance between computed and weighted focal lengths

Chapter 6

Conclusions

In this chapter, we summarize our work in pattern-based augmented reality, and describe interesting future avenues for research. We then conclude with a summary of the contributions that were made in this thesis.

6.1 Comparisons with Previous Work

The pattern-based augmented reality system that is most similar to ours is the ARToolkit technology [BILL01a, BILL01b, KATO00a, KATO00b]. Similar to our method, the ARToolkit augments virtual objects onto black and white planar patterns, but the technology cannot handle any partial pattern occlusions. Additionally, the ARToolkit performs a global search of the entire video frame for candidate patterns. By contrast, our localized corner-tracking approach provides better performance, especially at higher video resolutions.

An improvement to the ARToolkit technology was presented in [KATO00c], whereby multiple planar patterns and circular blobs were tracked simultaneously in order to account for occlusions. However, this required separate pose computations for each pattern, as well as prior knowledge of the relationship between the patterns in world coordinate space.

6.2 Future Work

6.2.1 Occlusion Between Real and Virtual Objects

Our current system allows for visual occlusion between multiple virtual objects due to the use of OpenGL and standard Z-buffer hardware. For real and virtual objects, however,

virtual objects are currently rendered in front of any real-world objects, regardless of the true occlusion relationships.

In order to handle visual occlusions between virtual objects and the real-world, an accurate model of the real-world is required. This model can be rendered transparently into the Z-buffer, causing subsequent renderings of virtual objects to only be drawn when they are closer than any real-world object [KLIN99].

While predefined CAD models of the real-world are possible when working in known AR environments, computer vision techniques are increasingly being used to recreate 3D information of a real-world scene for occlusion purposes [KLIN99]. A method that doesn't require 3D reconstruction is presented in [BERG97], which involves finding contours of objects in the real-world and determining whether they are in front or behind any virtual object. The contour information is then used to compute an occlusion mask that is used to determine real and virtual object ordering. A semi-automatic approach is presented in [LEPE00], but the method is not suited for real-time usage. The basic idea involves a user manually selecting occlusion areas in a small number of keyframes of a video sequence, and the algorithm automatically computes occlusions in intermediate views.

6.2.2 Virtual Lighting

Currently, all virtual objects are rendered at a fixed light intensity. Therefore any changes in the lighting conditions of the real environment have no effect on the appearance of virtual objects, which reduces the realism of the augmented scene. The difficulty with recreating lighting is being able to efficiently extract the positions of all light sources, as well as the reflective material properties of objects in the real world, from a real-time video sequence [KLIN99]. A method to extract a radiosity lighting model from 2D images is presented in [LOSC00], but the algorithm requires 3D information of the surrounding environment and it does not operate in real-time. A real-time algorithm is briefly discussed in [KLIN99], but it also requires an accurate 3D

description of the real environment that must be obtained either manually or by some other offline method.

6.2.3 Virtual Object Manipulation

One of the motivations for being able to handle partial occlusions of our planar patterns was to eventually allow manipulations of the virtual objects using hand gestures or special selection devices. Allowing a user to grasp a virtual object and move it or manipulate it in a natural manner would open the door to interesting applications for collaborative product design or entertainment. Additionally, issues such as virtual object physics (friction, gravity, etc.) and collision detection (between virtual-virtual and real-virtual objects) would further enhance the realism and immersiveness of an augmented environment. Some preliminary work that combines hand tracking with our augmented reality system is presented in [MCDO02].

6.2.4 Auditory and Haptic Devices

Another open area of research involves haptic devices, as discussed in Chapter 1. With a special glove, a user grasping a virtual object would experience force-feedback in his or her hands and fingertips, providing additional information regarding the shape and texture of the entity. A good discussion and implementation of a haptic device in an augmented reality environment is presented in [VALL98].

Auditory feedback would also benefit AR considerably, much in the same way as it enhances the real world. For example, humans regularly use sound to determine material properties of objects, such as glass, wood, concrete, etc. Providing similar information in an AR environment, in conjunction with haptic devices to simulate touch, would enhance realism considerably.

6.3 Summary

In this thesis we described a robust solution for vision-based augmented reality tracking that identifies and tracks, in real-time, known planar patterns consisting of a set of corners. The advantage in tracking corners is their robustness at a large range of distances, reliability under severe planar orientations, and tolerance of significant lighting changes or shadows.

Using the corner correspondences computed between the stored pattern and its projection in the image, it is possible to compute a 2D homography H between the two. With this computed homography we can perform uncalibrated 2D augmentations by placing any 2D picture in place of the pattern in the image. Additionally, the homography can be computed even when some of the corners are occluded.

From this computed homography it is possible to determine both the intrinsic and extrinsic parameters of the camera by an autocalibration process. This enables us to compute the full perspective camera projection matrix and thus perform augmentations of not just 2D objects, but also of perspective-correct 3D objects. Therefore, the augmentation system can automatically adjust to different types of cameras without having to go through a formal calibration step, as well as handle zoom lenses.

The design of the system was described, and experiments demonstrated the feasibility and reliability of the system under various situations, most significantly under partial pattern occlusion. This robustness, combined with the unique approach of using a homography to continuously calibrate the camera, should bring augmented reality one step closer to becoming a mass-market technology.

Appendix A

Subpixel Corner Finding

Corners remain relatively stable across a sequence of images in a video. For this reason, a number of corner finding algorithms [HARR88, TRUC98, BENE98] have been proposed in the computer vision literature for tracking purposes. The most popular is the Harris corner finder that computes a corner strength λ_p for each pixel **p** in a search window *W* of a grayscale video image [HARR88]. This is determined using a 2x2 symmetric matrix

$$C_p = \begin{pmatrix} \sum_{Q} E_x^{\,2} & \sum_{Q} E_x E_y \\ \sum_{Q} E_x E_y & \sum_{Q} E_y^{\,2} \end{pmatrix}$$

where *Q* is a (2N + 1)x(2N + 1) neighborhood of pixels around **p** (we use N=3), and E_x and E_y are respectively the x and y spatial image gradients around **p** using Sobel edge filters [TRUC98]. Geometrically, the two eigenvectors of C_p define edge directions at pixel **p**, and the two eigenvalues λ_1 and λ_2 define the edge strengths [TRUC98]. A strong corner is thus denoted by two large eigenvalues, where $\lambda_1 \geq \lambda_2$. A point is a corner if the smaller of the two eigenvalues $\lambda_p = \lambda_2$ is larger than some predetermined corner strength threshold λ_T.

Note that the λ_T value is dependent on the size of the chosen neighbourhood for **p**. Therefore, a histogram analysis of the λ_2 values of an entire image can be used to decide on a suitable threshold. The best threshold for corners will be located to the right of a large peak in the histogram [TRUC98]. Note that this should not be performed for each frame, since the corner finding algorithm is computationally expensive ($O(N^3)$) when performed on an entire image. For our system, a user-defined threshold is used.

Since the corner finder computes a λ_2 value for each pixel in *W*, the strongest corner will be denoted by the pixel with the largest λ_2 value. The problem with this method of corner

finding is the lack of subpixel precision on the computed corner coordinate. This could lead to corner features exhibiting full pixel jitter from frame to frame, which could affect homography stability in subsequent stages of the tracking system.

Fortunately, the λ_2 values provide a convenient facility which can be used to compute subpixel coordinates for the corners. Non-maximal suppression is first used to locate the strongest corner position **s** = (*x, y*) in the pixel neighbourhood [TRUC98]. A subpixel location is then computed based on weighting the corner strengths of the 4-connected neighbours as follows:

$$sub\_x = x + 0.5 + \frac{(\lambda_a - \lambda_b)}{2(\lambda_b - 2\lambda_s + \lambda_a)} \qquad sub\_y = y + 0.5 + \frac{(\lambda_c - \lambda_d)}{2(\lambda_d - 2\lambda_s + \lambda_c)}$$

where λ_i represents the corner strength of pixel *i* in Figure A.1.

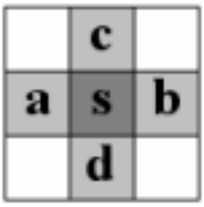


Figure A.1 – Four-connected neighbourhood for a corner pixel s.

Bibliography

[ABID00] M. A. Abidi, T. Chandra. "A New Efficient and Direct Solution for Pose Estimation Using Quadrangular Targets: Algorithm and Evaluation". IEEE Transactionson Pattern Analysis and Machine Intelligence. 17 (5), 1995. pp. 534-538.

[AUER99] Thomas Auer, Axel Pinz. "The Integration of Optical and Magnetic Tracking for Multi-User Augmented Reality". Computer & Graphics 23, 1999. pp. 805-808.

[AUER00] Thomas Auer, Axel Pinz. "Building a Hybrid Tracking System: Integration of Optical and Magnetic Tracking". Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, October 1999. pp. 13-22.

[AZUM97a] Ronald T. Azuma. "A Survey of Augmented Reality". *Presence: Teleoperators and Virtual Environments* 6, 4. August 1997. pp. 355-385.

[AZUM97b] Ronald T. Azuma. "Making Direct Manipulation Work in Virtual Reality". SIGGRAPH 97 Course Notes. August 1997.

[AZUM01] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. "Recent Advances in Augmented Reality". IEEE Computer Graphics and Applications. November / December 2001.

[BENE98] A. Benedetti, P. Perona. "Real-time 2D Feature Detection on a Reconfigurable Computer". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1998. pp. 586-593.

[BERG97] M. Berger. "Resolving Occlusion in Augmented Reality: A Contour Based Approach Without 3D Reconstruction". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1997. pp. 91-96.

[BILL01a] Mark Billinghurst, Hirokazu Kato, Ivan Poupyrev. "The MagicBook – Moving Seamlessly between Reality and Virtuality". *IEEE Computer Graphics and Applications*. May / June 2001.

[BILL01b] Mark Billinghurst, Hirokazu Kato, Ivan Poupyrev. "The MagicBook: A Transitional AR Interface". *Computer & Graphics*. Issue 25, 2001. pp. 745-753.

[BRAN99] Stefan Brantner, Thomas Auer, Axel Pinz. "Real-Time Optical Edge and Corner Tracking at Subpixel Accuracy". Computer Analysis and Image Processing. Proceedings of 8th International Conference on Computer Analysis of Images and Patterns CAIP '99. Springer, F. Solina, A. Leonardis (eds.), 1999. pp. 534-541.

[BROL01] Wolfgang Broll, Leonie Schafer, Tobias Hollerer, Doug Bowman. "Interface with Angels: The Future of VR and AR Interfaces". IEEE Computer Graphics and Applications. November / December 2001.

[DEBE99] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry: Algorithms and Applications, Second Edition. Springer Verlag, 1999.

[FISC81] M. A. Fischler and R. C. Bolles. "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". Communications of the ACM, 24(6), 1981. pp 381-395.

[FOLE90] Foley, J., van Dam, A., Feiner, S., Hughes, J. Computer Graphics: Principles and Practice. Addison-Wesley Publishing, Reading, Massachusetts, 1990.

[HARR88] C. Harris, M. Stephens. "A Combined Corner and Edge Detector". In Alvey Vision Conf, 1988. pp. 147-151.

[HOLL95] Holloway, R. "Registration Errors in Augmented Reality". PhD Thesis. UNC Chapel Hill, Department of Computer Science, August 1995.

[KANB01] Masayuki Kanbara, Naokazu Yokoya, Haruo Takemura. "A Stereo Vision-based Augmented Reality System with Marker and Natural Feature Tracking". Proceedings of the Seventh International Conference on Virtual Systems and Multimedia (VSMM'01), 2001. pp. 455-462.

[KATO00a] Hirokazu Kato, Mark Billinghurst, Ivan Poupyrev. *ARToolKit User Manual, Version 2.33.* Human Interface Technology Lab, University of Washington. November 2000.

[KATO00b] Hirokazu Kato, Mark Billinghurst. "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System". Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR) 1999. pp. 85-94.

[KATO00c] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, K. Tachibana. "Virtual Object Manipulation on a Table-Top AR Environment". Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR), 2000. pp. 111-119.

[KLIN99] Gudrun Klinker. "Augmented Reality: A Problem in Need of Many Computer Vision-Based Solutions". *Confluence of Computer Vision and Computer Graphics.* Kluwer Academic Publishers, 2000. pp 267-284.

[KUTU98] K. Kutulakos, J. Vallino. "Calibration-free Augmented Reality". IEEE Transactions on Visualization and Computer Graphics. 4(1), 1998.

[LEPE00] Vincent Lepetit, Marie-Odile Berger. "Handling Occlusion in Augmented Reality Systems: A Semi-Automatic Method". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000. pp. 225-230.

[LOSC00] Celine Loscos, George Drettakis, Luc Robert. " Interactive Virtual Relighting of Real Scenes". *IEEE Transactions on Visualization and Computer Graphics.* October-December 2000.

[MALI02a] Shahzad Malik, Gerhard Roth, Chris McDonald. "Robust Corner Tracking for Real-time Augmented Reality". In Proceedings of Vision Interface 2002.

[MALI02b] Shahzad Malik, Chris McDonald, Gerhard Roth. "Hand Tracking for Interactive Pattern-based Augmented Reality". In Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality, 2002.

[MCDO02] Chris McDonald, Shahzad Malik, Gerhard Roth. "Hand-based Interaction in Augmented Reality". In Proceedings of IEEE International Workshop on Haptic Audio Visual Environments and their Applications, 2002.

[MOLI01] Jose Molineros, Rajeev Sharma. "Real-Time Tracking of Multiple Objects Using Fiducials for Augmented Reality". Real-Time Imaging 7, 2001. pp. 495-506.

[NEUM99] Neumann, U., You, S., Cho, Y., Lee, J., Park, J. "Augmented Reality Tracking in Natural Environments". Proceedings of the first International Symposium on Mixed Reality (ISMR '99), pp 5-30, Yokohama, Japan. March 1999.

[OBER93] Denis Oberkampf, Daniel DeMenthon, Larry Davis. "Iterative Pose Estimation Using Coplanar Points". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1993. pp. 626-627.

[OHSH98] Ohshima, T., Satoh, K., Yamamoto, H., Tamura, H. "AR^{2}Hockey: A Case Study of Collaborative Augmented Reality". Proceedings of the IEEE International Symposium on Virtual Reality, 1998. pp. 268-275.

[REGE01] Holger Regenbrecht, Gregory Baratoff, Michael Wagner. "A Tangible AR Desktop Environment". *Computer & Graphics.* Issue 25, 2001. pp 755-763.

[REKI98] Jun Rekimoto. "Matrix: A Realtime Object Identification and Registration Method for Augmented Reality". *Proceedings of Computer Human Interaction*. 3rd Asia Pacific. pp. 63-68.

[RUSS95] John Russ. The Image Processing Handbook. CRC Press, Boca Raton, Florida. 1995.

[SHIM98] Ikuko Shimizu, Zhengyou Zhang, Shigeru Akamatsu, Koichiro Deguchi. "Head Pose Determination from One Image Using a Generic Model". Proceedings IEEE Third International Conference on Automatic Face and Gesture Recognition, April 1998. pp. 100-105.

[SEO00] Yongduek Seo, Ki-Sang Hong. "Weakly Calibrated Video-based Augmented Reality: Embedding and Rendering through Virtual Camera". Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR) 2000. pp. 129-136.

[SIMO99] Gilles Simon, Marie-Odile Berger. "Registration with a Zoom Lens Camera for Augmented Reality Applications". Proceedings of the 2nd IEEE International Workshop on Augmented Reality (IWAR), 1999. pp. 103-112.

[SIMO00] Gilles Simon, Andrew Fitzgibbon, Andrew Zisserman. "Markerless Tracking using Planar Structures in the Scene". Proceedings of the IEEE International Symposium on Augmented Reality (ISAR), 2000. pp. 120-128.

[SRIN97] M. A. Srinivasan, C Basdogan. "Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges". *Computers & Graphics*, Issue 21. pp 393-404.

[STAT00] A. State, G. Hirota, D. Chen, W. Garrett, M. Livingston. "Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking". http://www.cs.unc.edu/~us/hybrid.html

[STUR00] Peter Sturm. Algorithms for Plane-Based Pose Estimation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000. pp. 706-711 (vol. 1).

[TAMU01] H. Tamura, H. Yamamoto, A. Katayama. "Mixed Reality: Future Dreams Seen at the Border between Real and Virtual Worlds". IEEE Computer Graphics and Applications. November/December 2001.

[THOM00] Bruce Thomas, Ben Close, et al. "ARQuake: An Outdoor/Indoor Augmented Reality First Person Application". Proceedings of the 4th International Symposium on Wearable Computers, 2000. pp. 139-146.

[TRUC98] Emanuele Trucco, Alessandro Verri. *Introductory Techniques for 3D Computer Vision*. Prentice-Hall, 1998.

[VALL98] James R. Vallino. *Interactive Augmented Reality*. PhD Thesis, University of Rochester, Rochester, NY. November 1998.

[XU96] Gang Xu, Zhengyou Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach*. Kluwer Academic Publishers, 1996.

[YOU99a] You, S., Neumann, U., Azuma, R. "Hybrid Inertial and Vision Tracking for Augmented Reality Registration". Proceedings of IEEE Virtual Reality, 1999. pp. 260-267.

[YOU99b] You, S., Neumann, U., Azuma, R. "Orientation tracking for outdoor Augmented Reality Registration". IEEE Computer Graphics and Applications, Volume: 19 Issue: 6 , Nov.-Dec. 1999. pp. 36 -42

[ZHAN00] Zhengyou Zhang. "A Flexible New Technique for Camera Calibration". IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 22, No. 11, November 2000.

[ZISS98] Andrew Zisserman. "Geometric Framework for Vision I: Single View and Two-View Geometry". Lecture Notes, Robotics Research Group, University of Oxford.

[ZISS00] Andrew Zisserman, Richard Hartley. *Multiple View Geometry*. Cambridge University Press, 2000.